Министерство образования Республики Беларусь

Учреждение образования «[Институт информационных технологий](https://www.google.by/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&sqi=2&ved=0CDcQFjAC&url=http%3A%2F%2Fwww.iit-bsuir.by%2F&ei=unOvUpDmGemB4gSZ3IGQDQ&usg=AFQjCNHEr4the3QhkSSjmxbzcNJBZi5-Tg&sig2=GxV3hq7_8t34Csduk0HElg&bvm=bv.57967247,d.bGE&cad=rja) Белорусского государственного университета информатики и радиоэлектроники»

Факультет компьютерных технологий

Кафедра программного обеспечения информационных технологий

|  |
| --- |
|  |
|  |
|  |

**ОТЧЕТ**

**по преддипломной практике**

Место прохождения практики: ООО «Стоматологическое образование», г.Минск

Сроки прохождения практики: с 28.10.2019 по 23.11.2019

|  |  |  |
| --- | --- | --- |
|  |  | Студент группы 681061  А. Д. Малофеевский  Руководитель практики от БГУИР  И. Л. Калитеня |

Минск 2019

**СОДЕРЖАНИЕ**

[Введение 7](#_Toc29563442)

[1 Анализ предметной области 9](#_Toc29563443)

[1.1 Обзор области применения программного средства 9](#_Toc29563444)

[1.2 Обзор конкурирующих систем 10](#_Toc29563445)

[1.3 Постановка целей и задач на дипломное проектирование 15](#_Toc29563446)

[1.4 Входные данные 15](#_Toc29563447)

[1.5 Выходные данные 16](#_Toc29563448)

[2 Анализ требований к программному средству 17](#_Toc29563449)

[2.1 Разработка функциональной модели 17](#_Toc29563450)

[2.2 Диаграмма вариантов использования 19](#_Toc29563451)

[2.3 Схема работы программы 25](#_Toc29563452)

[3 Техническое проектирование программного средства 31](#_Toc29563453)

[3.1 Разработка диаграммы развёртывания 31](#_Toc29563454)

[3.2 Выбор решений и инструментов для разработки 32](#_Toc29563455)

[3.3 Разработка модели данных 34](#_Toc29563456)

[4 Тестирование программного средства 46](#_Toc29563457)

[4.1 Описание тестируемого стенда 46](#_Toc29563458)

[4.2 Функциональное тестирование 46](#_Toc29563459)

[4.3 Тестирование производительности 51](#_Toc29563460)

[4.4 Примеры ошибок 52](#_Toc29563461)

[5 Руководство пользователя 57](#_Toc29563462)

[6 Определение экономической эффективности 60](#_Toc29563463)

[6.1 Характеристики программного продукта 60](#_Toc29563464)

[6.2 Расчёт стоимостной оценки затрат программного продукта 60](#_Toc29563465)

[Заключение 65](#_Toc29563466)

[Список использованной литературы 66](#_Toc29563467)

[Приложение а исходный код приложения 67](#_Toc29563468)

**РЕФЕРАТ**

ПРОГРАММНОЕ СРЕДСТВО «ОНЛАЙН ГЕНЕРАТОР API ДЛЯ ПРИЛОЖЕНИЙ» НА ПЛАТФОРМЕ NODE JS: дипломный проект / А. Д. Малофеевский. – Минск : БГУИР, 2020, – п.з. – ХХХ с., чертежей (плакатов) – 6 л. формата А1.

Объектом исследования является клиент-серверное приложение для роботы с предустановленными и индивидуальными модулями с возможностью использования API, которое позволит использовать контент та сторонних клиентских приложениям.

Цель работы – разработка клиент-серверного приложения для упрощения развертывания серверного приложения основанного на микро-сервисах работающих на REST API, также в приложение позволит работать с пред установленными модулями и создавать индивидуальные модули

Разработка данного программного средства позволит упростить развертывание серверного приложения основанного на микро-сервисах и административное веб-приложение.

В ходе проектирования программного средства была разработана функциональная и инфологическая модель программного средства, схема алгоритмов основных процедур, схема программного средства, также было разработано руководство пользователя и были проанализированы аналоги разрабатываемого приложения. Клиент–серверное приложение было полностью протестировано. Определена экономическая эффективность приложения.

После тестирования и исправления всех ошибок, разработанное программное средство было использовано для реализации двух проектов, одном из проектов использовалось сгенерированное API для веб и мобильного приложения.

**СОДЕРЖАНИЕ**

[Введение 7](#_Toc29563233)

[1 Анализ предметной области 9](#_Toc29563234)

[1.1 Обзор области применения программного средства 9](#_Toc29563235)

[1.2 Обзор конкурирующих систем 10](#_Toc29563236)

[1.3 Постановка целей и задач на дипломное проектирование 15](#_Toc29563237)

[1.4 Входные данные 15](#_Toc29563238)

[1.5 Выходные данные 16](#_Toc29563239)

[2 Анализ требований к программному средству 17](#_Toc29563240)

[2.1 Разработка функциональной модели 17](#_Toc29563241)

[2.2 Диаграмма вариантов использования 19](#_Toc29563242)

[2.3 Схема работы программы 25](#_Toc29563243)

[3 Техническое проектирование программного средства 31](#_Toc29563244)

[3.1 Разработка диаграммы развёртывания 31](#_Toc29563245)

[3.2 Выбор решений и инструментов для разработки 32](#_Toc29563246)

[3.3 Разработка модели данных 34](#_Toc29563247)

[4 Тестирование программного средства 46](#_Toc29563248)

[4.1 Описание тестируемого стенда 46](#_Toc29563249)

[4.2 Функциональное тестирование 46](#_Toc29563250)

[4.3 Тестирование производительности 51](#_Toc29563251)

[4.4 Примеры ошибок 52](#_Toc29563252)

[5 Руководство пользователя 57](#_Toc29563253)

[6 Определение экономической эффективности 60](#_Toc29563254)

[6.1 Характеристики программного продукта 60](#_Toc29563255)

[6.2 Расчёт стоимостной оценки затрат программного продукта 60](#_Toc29563256)

[Заключение 65](#_Toc29563257)

[Список использованной литературы 66](#_Toc29563258)

[Приложение а исходный код приложения 67](#_Toc29563259)

**ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ**

В настоящей пояснительной записке применяются следующие определения и сокращения:

БД – база данных.

СУБД – система управления базами данных.

ПС – программное средство.

ПО – программное обеспечение.

ОС – операционная система.

ПК – персональный компьютер.

ЦП – центральный процессор персонального компьютера.

ОЗУ – оперативное запоминающее устройство.

ПЗУ – постоянное запоминающее устройство.

SSD – solid-state drive – твердотельный накопитель.

REST – representational state transfer – передача состояния представления

API – application programming interface – программный интерфейс приложения.

JS – javascript – мультипарадигменный язык программирования.

JSON – javascript object notation – простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером.

JWT – json web token – это открытый стандарт (RFC 7519) для создания ключа доступа, основанный на формате JSON.

SQL – structured query language – Структурированный язык запросов.

HTML – стандартизированный язык разметки документов в сети Интернет.

XML – расширяемый язык разметки.

XHTML – расширяемый язык гипертекстовой разметки.

DOM – document object model – объектная модель документа

URL – uniform resource locator – унифицированный указатель ресурса.

Codemods – скрипт разработанный компанией Facebook для обновления кода React JS из старой версии к новой.

Apache Bench – Используется для тестирования нагрузки на веб-сервер.

Apache2 – свободный веб-сервер. Apache является кроссплатформенным ПО, поддерживает операционные системы Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS.

Framework – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

UNIX-платформа – семейство переносимых, многозадачных и многопользовательских операционных систем, которые основаны на идеях оригинального проекта AT&T Unix.

ВВЕДЕНИЕ

Каждое современное приложение сегодня дает пользователям сети интернет возможность взаимодействовать с другими пользователями или приложением. Важно, чтобы приложение могло легко отправлять и получать данные между клиентами через ЛВС сеть или сеть интернет. Это также можно назвать способом обслуживания потребителей. Часть приложения, которая позволяет это сделать, называется API. Все основные приложения используют какой-то API для облегчения связи, API может быть внутренним или общедоступным, или также может быть коммерческим. Продажа доступа к API приложений - это большой бизнес, особенно если приложения предоставляют данные, которые не предоставляет никакой другой сервис или приложение. Поскольку API может легко содержать множество модульных частей, перед его созданием должено соблюдаться архитектурный стиль. Существует множество шаблонов для одного и того же, таких как Peer-to-Peer (P2P), REST, сервис-ориентированный, ориентированный на данные, управляемый событиями. Самым известным шаблон API из множества является REST. REST расшифровывается как передача состояния представления [1].

REST – это архитектурный стиль, на котором основано все современное программное обеспечение и веб-сервисы. Однако, несмотря на популярность REST, у него есть некоторые явные недостатки, которые нуждались в исправлении. [1].

Взаимодействие различных сервисов с использованием АPI, из новаторства превращается в рядовую операцию. Количество бесплатных и платных API уже исчисляется тысячами, и с каждым днем их число активно растет. Продажа удаленных запросов к своему новаторскому сервису может принести больше прибыли, чем распространение услуг через свою площадку [2].

Веб-приложение, а именно генератор API предоставит возможность пользователям реализовывать микро-сервисы в веб-интерфейсе позволяющие:

Обмен данными между различными приложениями вне зависимости от платформы на которой они разработаны;

* back-end сервер для реализации веб/мобильных-приложений с бизнес-логикой;
* микро-сервисы, работающие в реальном времени (блоки из соц. сетей, игры);
* микро-сервисы для глубокой аналитики.

Микро сервисная архитектура приложений в последние несколько лет используется как комплекс из нескольких микро-сервисных приложений. За частую встречаются такие приложения в которых используется от 4 микро-сервисов которые обледенены одним сервисом который контролирует работу остальных микро-сервисов.

На данный момент существует множество реализованных серверных приложений, работающих на API, но у подавляющего большинства есть проблемы. В основном проблемы заключаются в том, что приложение довольно старое, в них используются не актуальные архитектуры или устаревшие библиотеки, которые в свою очередь имеют уязвимости в безопасности. Также большая часть приложений являются платными или с закрытым исходным кодом что не дает возможность модернизировать приложение. За частую встречаются сервисы, которые работают не так как описано в документации или содержат критические ошибки в безопасности сессий или доступа к контенту.

Основными причинами, по которым стоит реализовать клиент-серверное приложение для генерации модулей API являются:

* простая и быстрая развёртка клиент-серверное приложение;
* масштабируемая архитектура, позволяющая модернизировать приложения;
* уменьшение нагрузки на серверное приложение за счёт микро-сервисной архитектуры.

База данных должна иметь чёткую структуру, позволяющую добавлять новые таблицы и вносить несущественные правки в старых, должна обладать свойством достаточности и полнотой индексов. Предпочтение следует отдать подсистеме СУБД PostgreSQL.

Темой данного дипломного проекта является программное средство онлайн генератор API для приложений на платформе Node JS.

Дипломный проект выполнен самостоятельно и проверен в системе «Антиплагиат» [3]. Процент оригинальности соответствует норме, установленной кафедрой.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор области применения программного средства

В сети интернет есть множество доступных серверных приложений, часть из них платная с закрытым исходным кодом, а также существует большое количество готовых серверных приложений, основанных на микро-сервисах, на ресурсе github.com. В сети интернет есть ресурсы, которые предоставляют веб-приложение, в котором можно настроить архитектуру приложения для работы интернет магазина, которая будет использовать технологию API для взаимодействия с пользователями. Или приложение, которое позволяет настроить работу CRM в веб-приложении также с применением технологии API для создания новых лидов или проводить полный цикл продажи через любое другое клиентское приложение. Но у всех этих приложений есть одна большая проблема, а именно данные приложения невозможно расширять, весь функционал, который предоставлен изначально невозможно расширить новыми микро-сервисами. По этому компании, которые разрабатывают собственное микро-сервисное приложение часто прибегают к таким ресурсам как github.com.

Проанализировав популярные решения на ресурсе github.com, можно прийти к следующему выводу, из тех приложений, которые были рассмотрены примерно 63% используют старые библиотеки в которых имеются дыры в безопасности или самописные библиотеки которые очень сложно поддерживать сторонним разработчикам, а также давно никем не обновлялись. 21% приложений на основе микро-сервисов не соответствуют описанию или документации представленной на ресурсе. Остальная часть приложений не доделанные, которые сложно взять за основу нового разрабатываемого приложения или приложения, которые использовались для создания видео уроков для начинающих программистов [4].

За счёт перечисленных проблем было решено разрабатывать собственное серверное приложения работающее на основе микро-сервисов API. Разработка собственного приложения решит сразу ряд проблем, таких как:

* масштабируемая архитектура приложения, она позволит легко разворачивать малые микро-сервисы или большие приложения позволяющее решать различные проблемы;
* использование необходимых модулей для решения современных задач;
* возможность дублирования или создания индивидуальных модулей;
* возможность модернизировать приложение;
* уменьшение нагрузки на приложение за счёт разделения модулей на отдельные микро-сервисы.

1.2 Обзор конкурирующих систем

Для создания программного средства, необходимо изучить аналоги и выделить их основные недостатки и преимущества, определить ведущие тенденции в данном направлении.

В качестве исследуемых аналогов были выбраны программные продукты, связанные с работой API, добавление индивидуальных модулей, работа с контентом, работа с аналитикой. Основным критерием для выбора служила актуальность данных программных средств, частота их использования.

Отличия разрабатываемого программного средства от выявленных аналогов.

Airship - это framework для Node JS, который помогает разрабатывать большие, масштабируемые и обслуживаемые API-серверы [5].

Основная идея проста, у каждого запроса есть своя модель у каждого ответа тоже есть модель. Важно, что на данный момент вся система даже ничего не знает о сети. Из-за этого система абстрактна, она просто обрабатывает указанные запросы и возвращает указанные ответы. Это дает возможность изменить сетевой протокол или даже прекратить использование системы в качестве веб-сервера и использовать его как часть локального приложения пользовательского интерфейса [5].

Преимущества системы:

* простая архитектура;
* автоматическую сериализацию / десериализацию моделей;
* возможность генерации схемы API;
* генерация документации на основе схемы;
* генерация простого клиента.

Недостатки системы:

* используется не реляционная БД;
* отсутствует какое-либо административное приложение;
* вся работа с приложением ведётся в коде.

Airship это оболочка для разработки серверного программного средства, оно обладает простой архитектурой. Данное программное средство позволяет быстро разработать серверное приложение на схемах API, но на данной платформе довольно сложно разработать сложную бизнес логику.

Вся не посредственная работа с framework ведётся в коде что для многих компаний является трудностью, а именно найме программиста, который должен будет разобраться с ПС или нанять сотрудника, который уже работал с ним.

На рисунке 1.1 представлена схема работы Airship framework.

|  |
| --- |
|  |

Рисунок 1.2 – Схема работы Airship framework

Hiboutik это веб-POS программное обеспечение, позволяет организовать работу интернет-магазина. Данное веб-приложение предоставляет разработчикам мощный API-интерфейс для интеграции данных в существующие системы [6].

Преимущества системы:

* быстрое и простое развёртывание;
* подходит для работы с интернет магазинами;
* подробная документация;
* не требует собственных или арендованных серверных машин.

Недостатки системы:

* узко направленная система;
* нет возможности модернизации приложения;
* приложение является платным.

Данное программное средство позволяет вести учет товаров, список клиентов и вести список продаж. С товарами предусмотрена базовые функции, такие как: прибытие товара на склад, наличие товара на склад, импорт и экспорт. На рисунке 1.2 представлена страница со списком товаров.

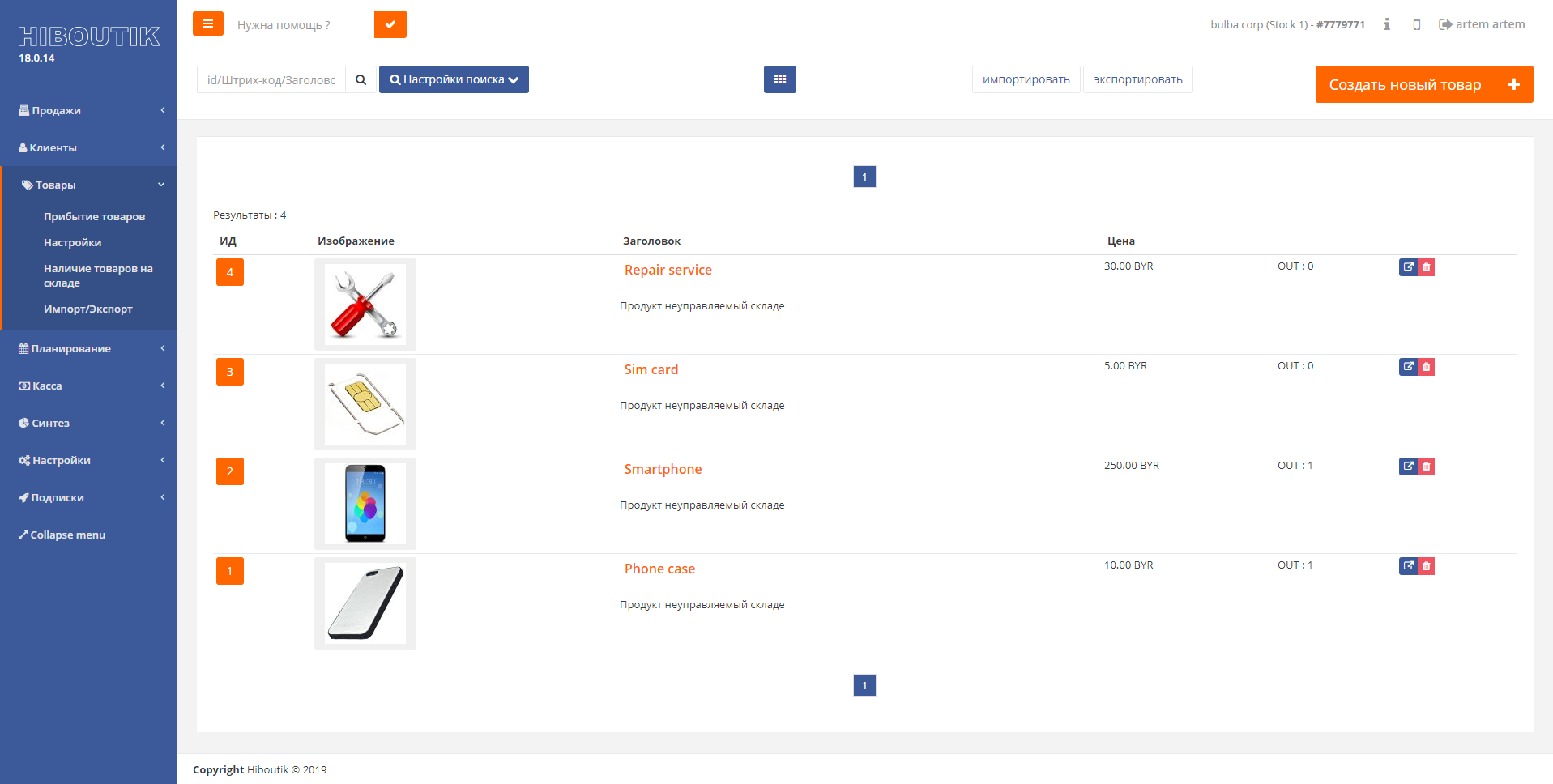


Рисунок 1.2 – Страница списка товаров

На странице продажи можно создать новую продажу или посмотреть продажи за определённый период или продажи по необходимому клиенту, на рисунке 1.3 представлена страница продаж.

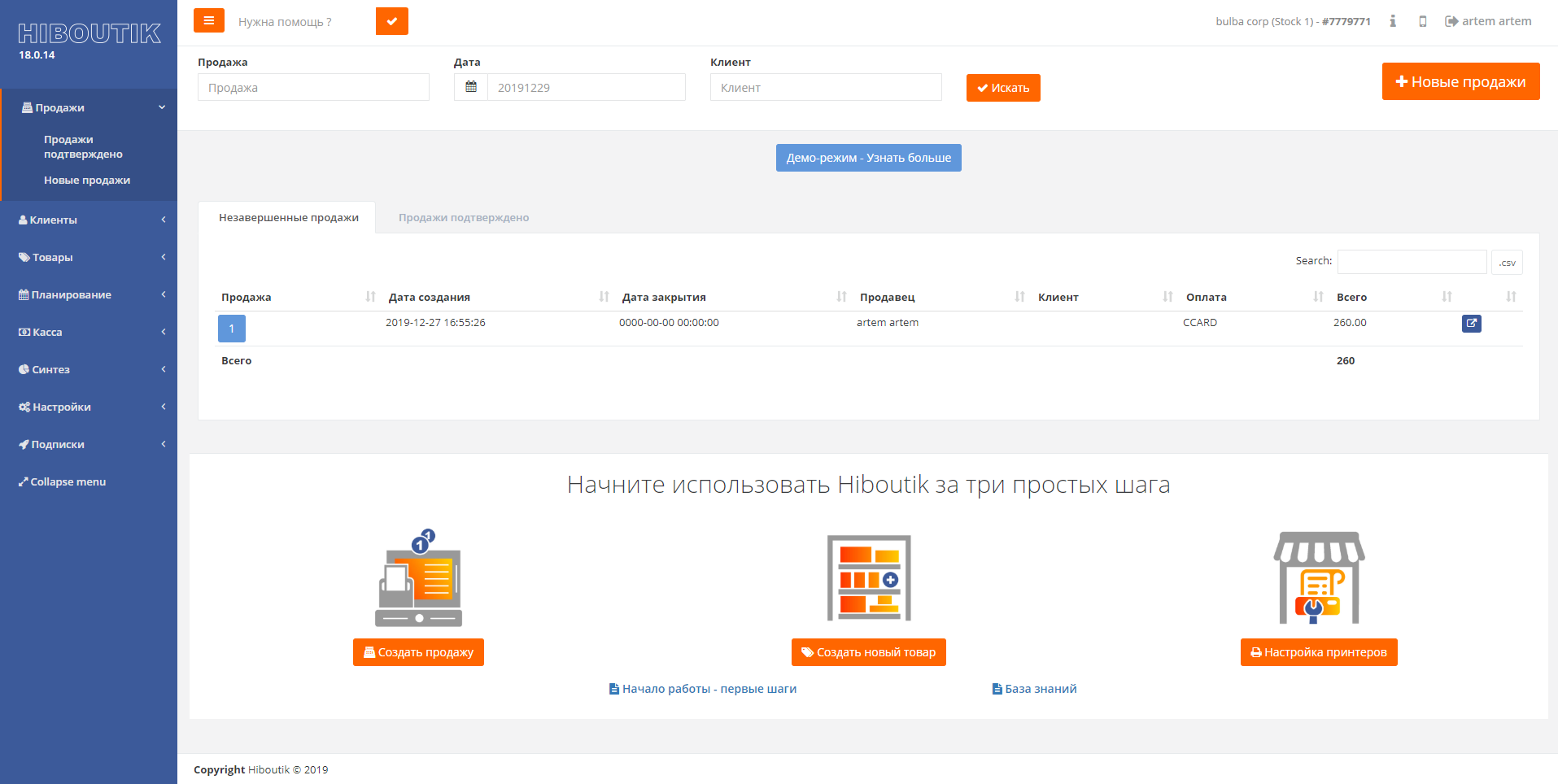


Рисунок 1.3 – Страница списка продаж

По итогу анализа программного средства Hiboutik было выяснено что данный продукт подходит только для ведения малого бизнеса и не годится для использования в крупных компаниях где процесс продаж или иная деятельность с клиентами не является стандартной или шаблонной по отношению к данным продуктам где работа с клиентами ведётся потоково.

AmoCRM - система управления взаимоотношениями с клиентами, позволяющая контролировать ход продаж и доступная в режиме online из любой точки мира [7].

Преимущества системы:

* быстрое и простое развёртывание;
* реализовано множество решений для работы с клиентами и продажами;
* простая в пользовании и не требует длительного обучения пользователя;
* присутствует пробный период;
* возможность управлять продажами и клиентами с помощью API;
* присутствует аналитика.

Недостатки системы:

* узко направленная система;
* нет возможности модернизации приложения;
* приложение является платным.

В данном программном продукте сделан упор на проведение сделок, в ПС ведется учёт клиентов, сделок, а также проводится аналитика по сделкам. Аналитику можно проводить по сделкам, по сотрудникам, звонкам. На рисунке 1.4 представлена страница анализа продаж.

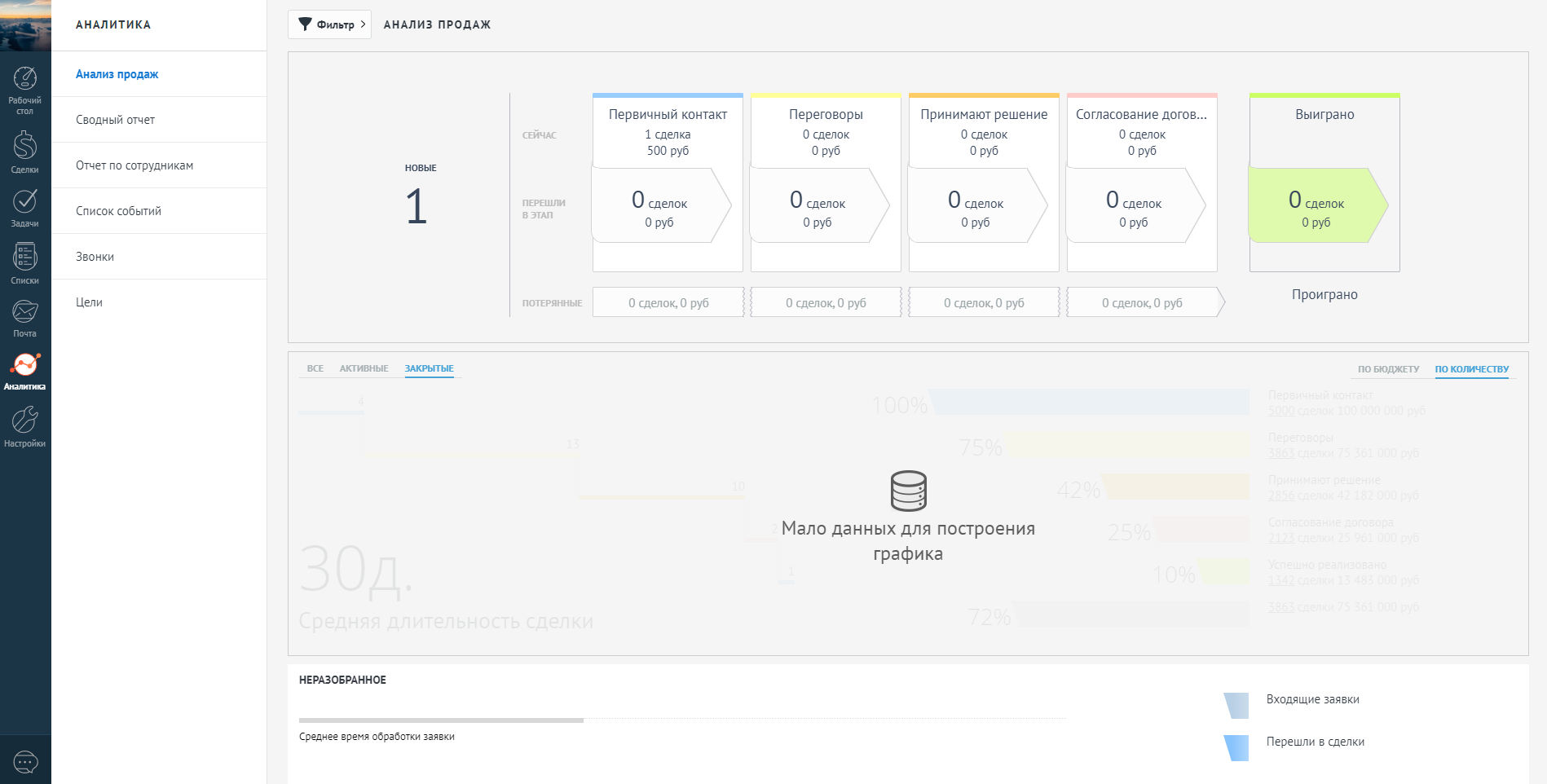


Рисунок 1.4 – Страница анализа продаж

У каждого добавленного сотрудника в программное средство индивидуальная страница сделок где он их проводит на данной странице он может отслужить на какой из стадий находиться сделка, также пользователь может создать новую сделку или завершить существующую. На рисунке 1.5 представлена страница сделок.

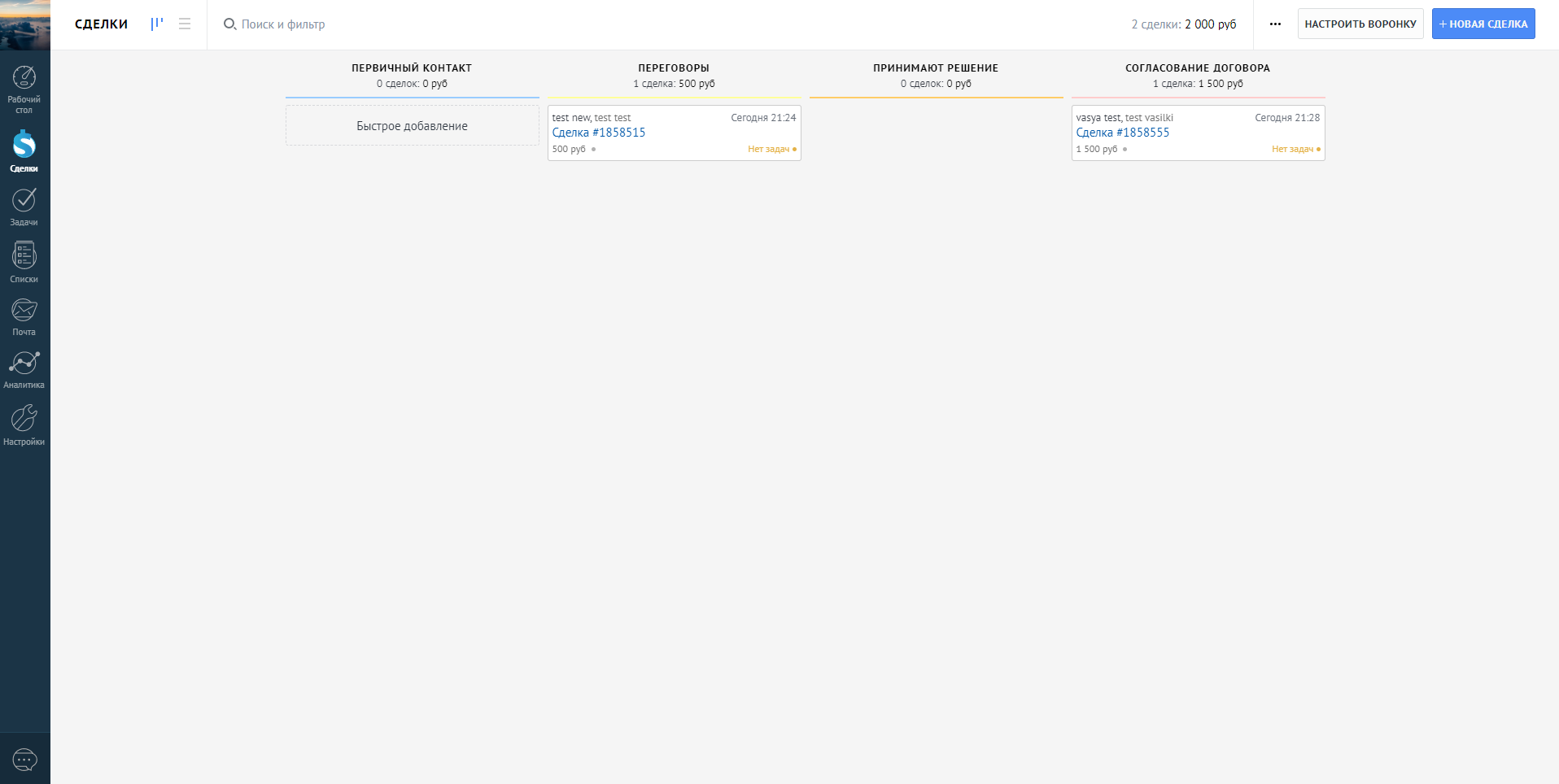


Рисунок 1.5 – Страница сделок

По итогу анализа данного ПС было выяснено что оно больше подходит для отдела продаж. AmoCRM является не плохим решение для проведения продаж, поддержки клиентов. Также реализовано множество не плохих решений, таких как встроенная почтовая служба для общения с клиентами, телефония, аналитика и постановка задачи для напоминания о намеченном действии с клиентом или поставить задачу на другого сотрудника. Также данное программное средство имеет не плохое встроенное API что позволяет начать работать с клиентом с первого его входа на сайт и не вносить в ручную сотрудникам данные клиента.

1.3 Постановка целей и задач на дипломное проектирование

Назначение программного средства является автоматизация и упрощение развёртывание серверного приложения основанном на микро-сервисах, работающих на REST API.

Программа должна обеспечивать выполнение перечисленных ниже функций:

* простое развертывание серверного приложения;
* масштабирование микро-сервисов;
* регистрация пользователей;
* надёжное шифрование паролей;
* создание сессии для пользователей с использованием JWT;
* защита сессии от не санкционированного доступа из вне;
* пользовательская настройка прав доступа к админской и внешней части приложения;
* выдача прав доступа пользователям;
* защита CROS от не санкционированного доступа к API проекта;
* аналитика просмотра контента уникальными пользователями;
* хранение и рассылка почтовых шаблонов(оповещений) по электронной почте;
* аналитика по рассылке;
* создание индивидуальных модулей;
* ограничение доступа к пользовательскому контенту;
* обработка и хранение файлов на сервере;
* сервисы, работающие в реальном времени (лента новостей, комментарии, чаты).

1.4 Входные данные

Для входа в систему необходимо ввести e-mail и пароль, после чего пользователь сможет выполнять свои задачи.

Входной информацией будут является:

* данные для регистрации и авторизации;
* данные о пользователе;
* пользовательские настройки;
* загрузка файлов;
* добавление, редактирование, удаление и просмотр предустановленных и индивидуальных модулей;
* добавление, редактирование, удаление и просмотр записей в предустановленных и индивидуальных модулях;
* методы запросов: GET, POST, PUT, DELETE.

1.5 Выходные данные

В качестве выходных данных будет выступать:

* сессия и JWT;
* пользовательская информация;
* оповещения пользователя;
* данные предустановленных и индивидуальных модулей;
* файлы;
* аналитика;
* ошибки.

В результате проведенного анализа предметной области, был проведён обзор области применения программного средства. Исходя из обзора были проанализированы ресурсы с готовыми решениями программных средств позволяющие работать с API, в результате чего представлена статистка по приложениям, по которой можно сделал вывод что реализация собственного приложения, работающего на микро-сервисах правильное решение которое позволит решить множество проблем готовых решений. Также были рассмотрены конкурирующие системы по итогу которых были описаны положительные и отрицательные стороны систем. По итогам анализа была поставлена цель и задачи, которые необходимо выполнить в дипломном проекте. Также описаны входные и выходные данные.

2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ

2.1 Разработка функциональной модели

Функциональная модель программного средства представлена диаграммами А-0 и А0.

Для формализации и описания процесса разработки ПО используется методология функционального моделирования и графическая нотация IDEF0. Отличительной особенностью данной нотации является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

Методология IDEF0 используется благодаря простой и понятной для понимания графической нотации. Главное место в методологии отводится диаграммам. На диаграммах отображают функции системы посредством геометрических прямоугольников, а также имеющиеся связи между функциями и внешней средой [8].

IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес–процессов. Отличительной особенностью IDEF0 является акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность [8].

Стандарт IDEF0 представляет организацию как набор модулей, существует правило – наиболее важная функция находится в верхнем левом углу, кроме того есть правило стороны:

* стрелка входа всегда приходит в левую кромку активности;
* стрелка управления – в верхнюю кромку;
* стрелка механизма – нижняя кромка;
* стрелка выхода – правая кромка [9].

Каждый функциональный блок в рамках единой рассматриваемой системы должен иметь свой уникальный идентификационный номер. Блок выглядит как «чёрный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня. Также для того чтобы быть правильно понятым, существуют словари описания активностей и стрелок. В этих словарях можно дать описания того, какой смысл вы вкладываете в данную активность либо стрелку.

Диаграммы А-0 представлена на рисунке 2.1.

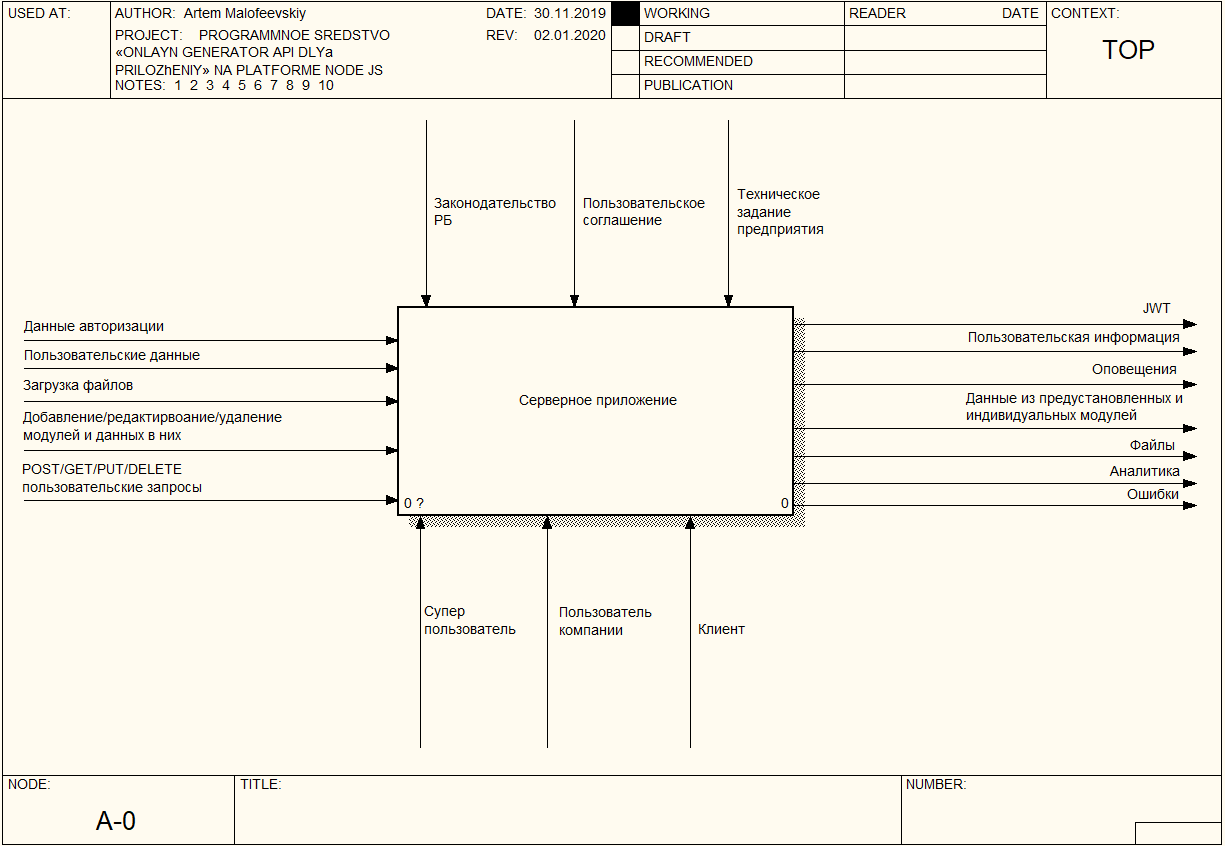


Рисунок 2.1 – Диаграмма А-0

На диаграмме А–0 (IDEF0) изображены выходные данные – JWT, пользовательская информация, оповещения, данные из предустановленных и индивидуальных модулей, файлы, аналитика. Входные данные – данные авторизации, пользовательские данные, загрузка файлов, добавление / редактирование / удаление индивидуальных модулей и данных в них, POST / GET / PUT / DELETE пользовательские методы запросов.

Используемые методы запросов:

* метод запроса: POST – предназначен для запроса, при котором [веб-сервер](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) принимает данные;
* метод запроса: GET – предназначен для получения информации от [веб-сервер](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80);
* метод запроса: PUT – предназначен для запроса, при котором [веб-сервер](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) принимает данные и заменяет их в существующей записи;
* метод запроса: DELETE – предназначен для запроса, при котором [веб-сервер](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) удаляет выбранную запись.

В верхней части диаграммы отображаются основный законы, которыми необходимо руководствоваться при создании программного средства.

Диаграмма А0 представлена на рисунке 2.2.

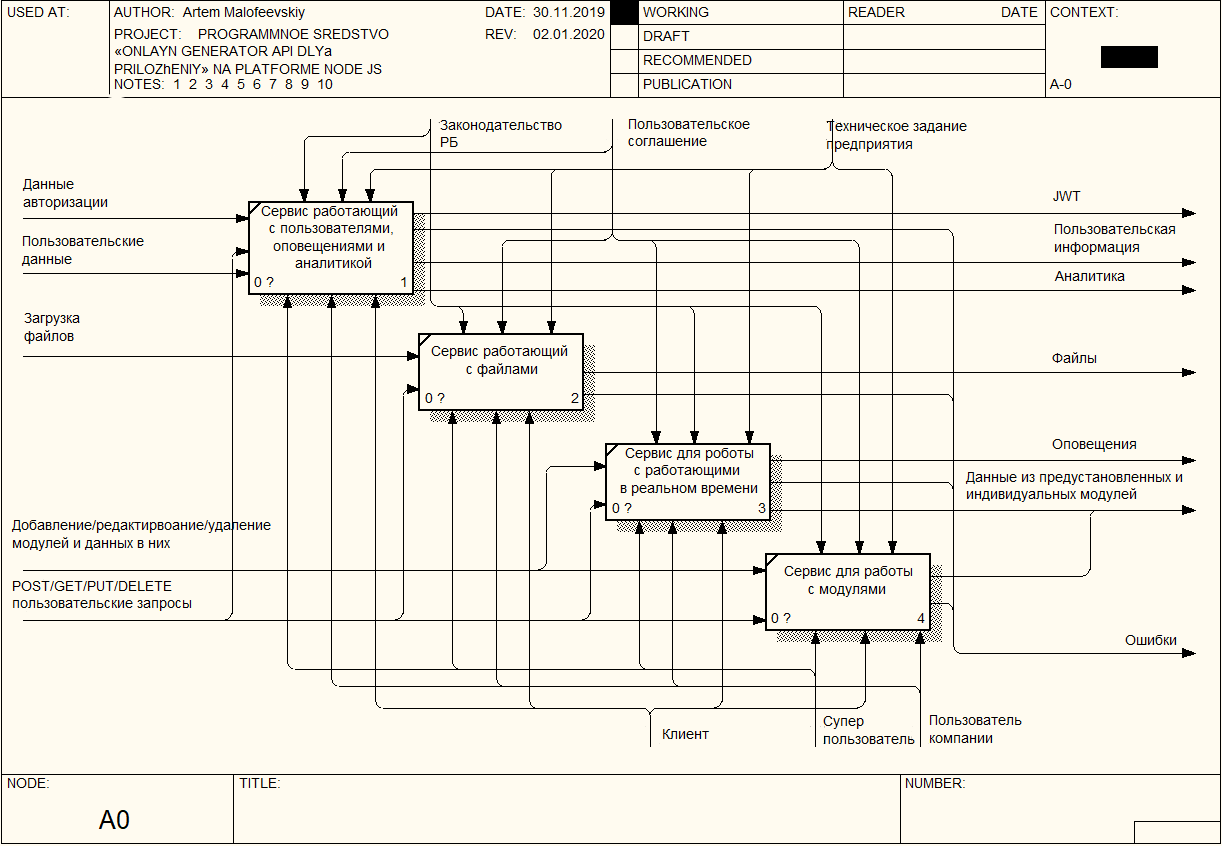


Рисунок 2.2 – Диаграмма А0

На диаграмме А0 (декомпозиция IDEF0) изображено подробное описание логики функционирования модулей программного средства и их взаимодействие с входными и выходными данными.

Модель IDEF0 используется при организации бизнес-процессов и проектов, основанных на моделировании всех процессах как административных, так и организационных.

2.2 Диаграмма вариантов использования

Диаграмма вариантов использования (use case) – это исходное концептуальное представление или концептуальная модель системы в процессе ее проектирования и разработки [10].

Диаграмма вариантов использования – диаграмма, на которой изображаются отношения между актерами и вариантами использования.

Вариант использования – внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами .

Актер – согласованное множество ролей, которые играют внешние сущности по отношению к вариантам использования при взаимодействии с ними.

В программном средстве присутствует два актера супер пользователь, пользователь компании, клиент. Супер пользователю предоставлены права на выполнение всех доступных функций:

* просмотр, добавление, редактирование, удаление пользователей;
* просмотр, добавление, редактирование, удаление пользовательской информации;
* просмотр, добавление, редактирование, удаление модулей;
* просмотр, добавление, редактирование, удаление записей модуля;
* просмотр, добавление, удаление файлов.

Пользователю компании обладает почти всеми вышеперечисленные функции за исключением тех, на которые у пользователя нет прав доступа.

Клиенту предоставлены права на выполнение следующих функций:

* авторизация и регистрация;
* просмотр, добавление, редактирование, удаление пользовательской информации;
* просмотр записей модуля;
* добавление записей в модуль (если есть права доступа).

Диаграмма вариантов использования для клиент-серверного приложения представлена на рисунке 2.7.

|  |
| --- |
|  |

Рисунок 2.7 – Диаграмма вариантов использования

Диаграмма вариантов использования описывает взаимоотношения и зависимости между группами вариантов использования и действующими лицами, участвующими в процессе.

**2.2.1** Описание вариантов использования

Описанные варианты использования «Регистрации», «Добавление пользовательской информации», «Загрузка файлов», «Создание индивидуальных модулей», «Просмотр записей модуля», «Просмотр аналитики». Для удобства рассмотрения представлены в табличном формате.

В таблице 2.1 представлено описание варианта «Регистрации».

Таблица 2.1 – Описание варианта «Регистрации»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Описание | Актёры | Предусловия | Постусловия |
| Регистрация | Описывает регистрацию в системе. Используется, когда пользователь хочет зарегистрироваться в системе. | Применимо ко всем актёрам | Отсутствуют | При успешном выполнении происходит вход в систему. При неудачном –состояние системы не изменится |

В таблице 2.2 представлено описание варианта «Добавление пользовательской информации».

Таблица 2.2 – Описание варианта «Добавление пользовательской информации»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Описание | Актёры | Предусловия | Постусловия |
| Добавление пользовательской информации | Описывает добавление пользовательской информации в системе. Используется, когда пользователь хочет добавить пользовательскую информацию в систему. | Применимо ко всем актёрам | Отсутствуют | При успешном выполнении происходит сохранение данных в системе. При неудачном –состояние системы не изменится |

В таблице 2.3 представлено описание варианта «Загрузка файлов».

Таблица 2.3 – Описание варианта «Загрузка файлов»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Описание | Актёры | Предусловия | Постусловия |
| Загрузка файлов | Описывает загрузку файлов в систему. Используется, когда пользователь хочет загрузить новый файл в систему. | Применимо ко всем актёрам | Проверка наличия прав доступа | При успешном выполнении происходит сохранение файла или файлов в систему. При неудачном –состояние системы не изменится |

В таблице 2.4 представлено описание варианта «Создание индивидуальных модулей».

Таблица 2.4 – Описание варианта «Создание индивидуальных модулей»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Описание | Актёры | Предусловия | Постусловия |
| Создание индивидуальных модулей | Описывает создание индивидуальных модулей в системе. Используется, когда пользователь хочет создать индивидуальный модуль в системе. | Супер пользователь, Пользователь компании | Проверка наличия прав доступа | При успешном выполнении происходит сохранение модуля в систему. При неудачном –состояние системы не изменится |

В таблице 2.5 представлено описание варианта «Просмотр записей модуля».

Таблица 2.5 – Описание варианта «Просмотр записей модуля»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Описание | Актёры | Предусловия | Постусловия |
| Просмотр записей модуля | Описывает просмотр записей модуля в системе. Используется, когда пользователь хочет просмотреть информацию из модуля в системе. | Применимо ко всем актёрам | Проверка наличия прав доступа | При успешном выполнении происходит вывод информации в приложение. При неудачном –состояние системы не изменится |

В таблице 2.6 представлено описание варианта «Просмотр аналитики».

Таблица 2.6 – Описание варианта «Просмотр аналитики»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Описание | Актёры | Предусловия | Постусловия |
| Просмотр аналитики | Описывает просмотр аналитики в системе. Используется, когда пользователь хочет просмотреть аналитику в системе. | Супер пользователь, Пользователь компании | Проверка наличия прав доступа | При успешном выполнении происходит вывод информации в приложение. При неудачном –состояние системы не изменится |

В таблице 2.7 представлено описание варианта «просмотр, редактирование и добавление меток в предустановленном модуле «Метки»».

Таблица 2.7 – Описание варианта «просмотр, редактирование и добавления меток в предустановленном модуле «Метки»».

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Описание | Актёры | Предусловия | Постусловия |
| Просмотр и редактирование меток в предустановленном модуле «Метки» | Описывает просмотр, редактирование и добавление меток в систему. Используется, когда пользователь хочет работать с метками. | Супер пользователь, Пользователь компании | Проверка наличия прав доступа | При успешном выполнении происходит вывод существующих записей из модуля в приложение. При неудачном –состояние системы не изменится |

В таблице 2.8 представлено описание варианта «добавление записи в модуль «Записи»».

Таблица 2.7 – Описание варианта «добавление записи в модуль «Записи»».

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Описание | Актёры | Предусловия | Постусловия |
| добавление записи в модуль «Записи» | Описывает добавление записи в модуль «Записи». Используется, когда пользователь хочет добавить новую запись в модуль «записи». | Супер пользователь, Пользователь компании | Проверка наличия прав доступа | При успешном выполнении происходит запись в базу данных вывод информирующего уведомления и добавляется запись в приложение. При неудачном –состояние системы не изменится |

2.3 Схема работы программы

При проектировании программного средства была разработана функциональная модель. На основе этой модели были построены алгоритмы. Общая схема работы программного средства представлена на рисунке 2.3.

|  |
| --- |
|  |

Рисунок 2.3 – Общая блок-схема работы программного средства

Представленный алгоритм представляет общую схему работы приложения. Перед запуском приложения необходимо запустить сервер, после чего будет доступно веб-приложение в котором ведётся всё основная работа. Чтобы выполнять работу в приложении необходимо авторизоваться, после чего загрузиться главная страница, на главной странице будет отображены разделы, к которым у пользователя есть доступ.

**2.3.1** Алгоритм регистрации на стороне сервера

Схема работы алгоритма регистрации на стороне сервера представлена на рисунке 2.4.

|  |
| --- |
|  |

Рисунок 2.4 – Блок-схема работы алгоритма регистрации на стороне сервера

В представленном алгоритме отображена схема работы регистрации на серверном приложении, веб-приложение отправляет запрос на сервер после чего приложение считывает данные, проводит проверку данных, проверяет существует ли пользователь с таким же логином, если пользователь не существует, то приложение шифрует пароль, записывает данные в БД, генерирует сессию и JWT и возвращает данные пользователю.

**2.3.2** Алгоритм проверки сессии и JWT на стороне сервера

Схема работы алгоритма проверки сессии и JWT представлена на рисунке 2.5.

|  |
| --- |
|  |

Рисунок 2.5 – Блок-схема проверки алгоритма генерации сессии и JWT

Представленная схема представляет работу модуля проверки сессии и JWT. При каждом запросе на сервер модуль автоматически проверяет есть ли у в теле запроса JWT, если есть считываем JWT и проверяем активен ли он, если активен, то получаем информацию о пользователе и правах доступа, после чего проверяется подлинный ли пользователь использует JWT, если да, то пользователю или в следующий модуль поступает информация о пользователе и о группе прав.

**2.3.3** Алгоритм проверки уровня доступа на стороне сервера

Схема работы алгоритма работы проверки уровня доступа пользователя на серверном приложении представлена на рисунке 2.6.

|  |
| --- |
|  |

Рисунок 2.7 – Блок-схема работы проверки уровня доступа пользователя на серверном приложении

**2.3.4** Алгоритм работы с модулями

Схема работы алгоритма работы с предустановленными и индивидуальными модулями представлена на рисунке 2.7.

|  |
| --- |
|  |

Рисунок 2.7 – Блок-схема работы с предустановленными и индивидуальными модулями

В представленном алгоритме отображена работа модулей, изначально нужно перейти на нужную страницу в веб-приложении, после чего проверятся уровень доступа и если доступ есть, то страница модуля загрузиться, иначе выводиться ошибка уровня доступа. После чего пользователь может добавлять, редактировать и удалять записи из модуля, после заполнения формы для записи она проходит проверку данных на стороне клиента в положительном результате данные отправятся на сервер, сервер также проведёт проверку данных и в случае успеха данные будут записаны в БД, в отрицательном результате пользователю выведет соответствующую ошибку.

В результате анализа требований к программному средству была разработана и описана функциональная модель, которая представлена в виде диаграмм A-0 и A0. Также была разработана диаграмма вариантов использования (use case) в которой описаны алгоритмы использования программного средства пользователями с разными уровнями доступа, также были описаны отдельно некоторые варианты использования. Были также реализованы и описаны схемы работы клиентского веб-приложения и алгоритмы, которые выполняются на стороне сервера. Все схемы были описаны и проиллюстрированы на рисунках.

3 ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Разработка диаграммы развёртывания

Программное средство имеет клиент-серверную архитектуру – система состоит из пяти компонентов – клиента и микро-сервисов. Клиент – это совокупное название потребительского (пользовательского) приложения, а микро-сервисы – это служебная часть, скрытая от пользователя. Структура системы представлена на диаграмме развертывания, рисунок 3.1.

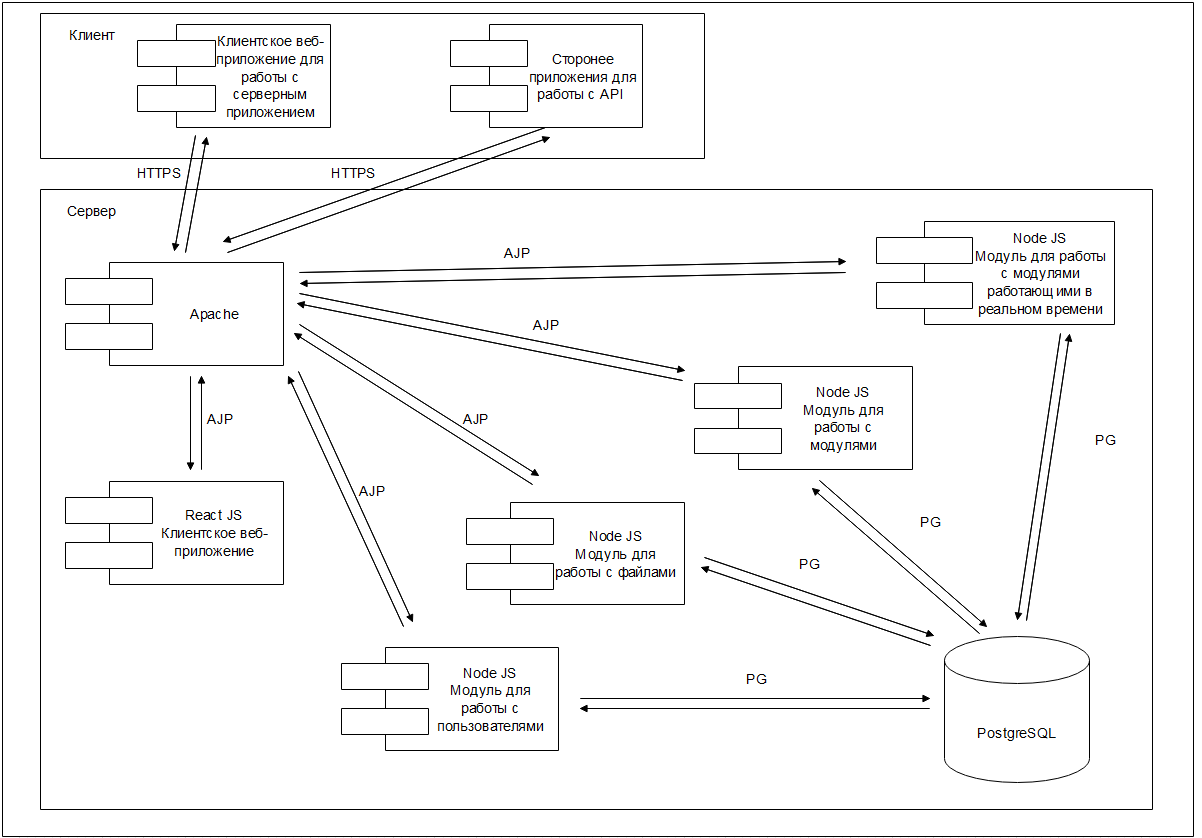


Рисунок 3.1 – Диаграмма развёртывания

Диаграммы развертывания отображают физическое развертывание артефактов (программных компонентов) на узлах (аппаратных компонентах), поэтому возможно увидеть, какие части программного обеспечения живут на каких аппаратных компонентах и как эти артефакты на узлах взаимосвязаны.

На диаграмме видно, что основная часть информации системы будет обрабатываться на сервере в микро-сервисных приложениях. Клиентское-приложение будет отвечать за визуальную часть.

Серверу передается запрос, который в свою очередь после обработки передается на СУБД. Далее из базы данных на сервер поступают данные, после чего данные передаются клиенту.

3.2 Выбор решений и инструментов для разработки

Для реализации данного микро-сервисного приложения было решено использовать платформу Node JS, а для реализации веб-приложения будет использоваться React JS.

Node JS – программная платформа, основанна на движке V8 (транслирующем JavaScript в машинный код), преобразовывает JavaScript из узко направленного языка в язык общего назначения. Node JS проста и интуитивно понятна даже для начинающих разработчиков. При этом Node.JS позволяет работать с серверными технологиями, реализовывать интерактивную работу с использованием компьютерных мощностей пользователей. В числе прочего, эта платформа позволяет запускать код из командной строки любой из распространенных ОС [11].

Преимущества использования Node JS:

* простой и известный язык разработки JavaScript;
* богатая стандартная библиотека;
* большое количество внешних библиотек и готовых модулей;
* движок V8.

Для реализации микро-сервисов на платформе решено использовать следующие модули:

1. express – веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений;
2. express-promise-router – простая оболочка для маршрутизатора Express 4, которая позволяет промежуточному ПО возвращать ответы на запросы.
3. cross-env – служит для определения переменных в запуске приложения;
4. body-parser – считывает и получает входящие данные в теле запросов в промежуточном программном обеспечении перед обработчиками;
5. cookie-parser – считывает и получает входящие данные запросов в теле cookie в промежуточном программном обеспечении перед обработчиками;
6. path – предоставляет утилиты для работы с путями файлов и каталогов;
7. mime-types – служит для работы с типами файлов;
8. multer – необходим для сохранения файлов;
9. session-express – требуется для работы с сессиями;
10. passport – библиотека для работы с сессиями и JWT;
11. passport-jwt –дополнительная библиотека для работы модуля passport с JWT;
12. passport-local – служит для хранения сессии
13. pg – необходим для подключения к базе данных PostgreSQL и заботы с ней;
14. crypto – библиотека обеспечивает криптографическую функциональность;
15. jsonwebtoken – служит для генерации JWT;
16. joi – необходим для валидации запросов.

React JS – это библиотека JavaScript, исходный код которой был открыт Facebook. Этот framework подходит для создания веб-приложений, где данные могут меняться на регулярной основе [12].

Преимущества использования React JS:

* легок в изучении, ввиду простоты его синтаксиса;
* высокий уровень гибкости;
* виртуальная DOM, которая позволяет упорядочивать документы форматов HTML, XHTML или XML в дерево, которое подходит веб-браузерам для анализа различных элементов веб-приложения;
* в сочетании с ES6/7 React JS может работать при высоких нагрузках;
* связывание данных от больших к меньшим. Поток данных, при котором дочерние элементы не могут влиять на родительские данные;
* простая миграция между версиями, Facebook предоставляет codemods для автоматизации большей части этого процесса.

PostgreSQL предоставляет множество различных возможностей, достаточно надежна и имеет хорошие характеристики по производительности. Она работает практически на всех UNIX-платформах. Ее можно применять на Windows NT Server и Windows Server, а для разработки годятся такие системы Microsoft для рабочих станций, как ME. Кроме того, PostgreSQL свободно распространяется и имеет открытый исходный код [13].

PostgreSQL отличается от многих других СУБД. Помимо того, что она обладает всеми возможностями других СУБД, а также обладает дополнительными возможностями.

Перечень функциональных возможностей PostgreSQL:

* транзакции;
* вложенные запросы;
* представления;
* ссылочная целостность - внешние ключи;
* сложные блокировки;
* типы, определяемые пользователем;
* наследственность;
* правила;
* проверка совместимости версий.

SQL – язык, который дает возможность создавать и работать в реляционных базах данных, являющихся наборами связанной информации, сохраняемой в таблицах.

Информационное пространство становится более унифицированным. Это привело к необходимости создания стандартного языка, который мог бы использоваться в большом количестве различных видов компьютерных сред. Стандартный язык позволит пользователям, знающим один набор команд, использовать их для создания, нахождения, изменения и передачи информации - независимо от того, работают ли они на персональном компьютере, сетевой рабочей станции, или на универсальной ЭВМ [14].

3.3 Разработка модели данных

Модель данных представлена инфологической моделью. Инфологическое моделирование выполняется с целью обеспечения естественных для человека способов представления и сбора информации, которая будет храниться в создаваемой БД.

Поэтому инфологическая модель данных строится в соответствии с естественным языком, который невозможно использовать в чистом виде в виду сложности обработки текстов с помощью компьютера и неоднозначности естественного языка [15].

Инфологическая модель – это потоки информации, сущности и связи данной области. В такой модели указываются связи между сущностями данной предметной области.

Сущность – это любой объект, отличающийся от другого, информацию о котором необходимо сохранить.

Связь – это ассоциирование нескольких сущностей с целью отыскания одних из них по значениям других.

База данных может содержать неограниченное количество сущностей и такое же количество связей между ними, что определяет сложность инфологических моделей.

Атрибут – это характеристика сущности. Его наименование должно быть уникальным.

Ключ представляет собой минимальное количество атрибутов, с помощью которого можно отыскать необходимый экземпляр сущности.

Связи между сущностями:

* один–к–одному (1:1);
* один–ко–многим (1:М);
* многие–ко–многим (М:М).

Цель инфологического моделирования – обеспечить оптимальные способы сбора и представления информации, хранимой в базе данных.

При работе над проектом были созданы таблицы, которые содержат следующие поля:

* таблица «users»: «код пользователя», «логин», «хеш пароля», «хеш соли», «дата регистрации», «активный хеш ключ», «текст статуса», «статус»;
* таблица «profiles»: «код профиля», «код пользователя», «имя», «фамилия», «дата рождения», «код картинки пользователя»;
* таблица «level\_access»: «код уровня доступа», «название», «заголовок», «описание», «статус»;
* таблица «users\_access»: «код уровня доступа пользователя», «код пользователя», «код уровня доступа»;
* таблица «module\_access»: «код уровня доступа модуля», «код пользователя», «код модуля»;
* таблица «image\_base»: «код картинки», «код пользователя», «оригинальное название», «хеш названия», «путь к картинке», «дата загрузки», «статус»;
* таблица «load\_images»: «код загрузки картинки», «код картинки», «код пользователя», «опциональные настройки»;
* таблица «users\_images»: «код пользователя картинки», «код пользователя», «код картинки», «опция»;
* таблица «files\_base»: «код файла», «код пользователя», «оригинальное название», «хеш названия», «путь к файлу», «дата загрузки», «статус»;
* таблица «load\_files»: «код загрузки файла», «код файла», «код пользователя», «опциональные настройки»;
* таблица «users\_files»: «код пользователя файла», «код пользователя», «код файла», «опция»;
* таблица «type\_variables»: «код типа переменной», «название», «заголовок», «описание», «статус»;
* таблица «schema\_varialbles»: «код схемы переменной», «название», «заголовок», «описание», «схема», «статус»;
* таблица «variables»: «код переменной», «код схемы переменной», «название», «заголовок», «описание», «контент», «статус»;
* таблица «type\_modules»: «код типа модуля», «название», «заголовок», «описание», «статус»;
* таблица «modules»: «код модуля», «код типа модуля», «название», «заголовок», «описание», «шаблон модуля», «схема шаблона модуля», «контент», «настройки», «статус»;
* таблица «rows\_modules»: «код строки модуля», «код модуля», «дата создания», «дата обновления», «код пользователя», «статус»;
* таблица «content\_rows\_module»: «код контента строки модуля», «код строки модуля», «код переменной», «контент».

Диаграмма сущность–связь будет представлена на рисунке 3.2.

|  |
| --- |
|  |

Рисунок 3.2 – Диаграмма сущность–связь

Диаграмма сущность связь предоставляет полную информацию о структуре базы данных программного средства в графическом виде.

Данные в таблицах хранятся в определённом формате, который называется типом данных. Типы данных могут быть числовыми, для работы с датой и временем, составные, бинарные и символьными.

Размер поля определяется для текстовых полей. Он показывает максимальное количество символов в поле. База данных соответствует реляционной модели данных, где каждый выделенный в ходе проектировании сущности соответствует таблица. Структура базы данных разрабатываемого программного средства включает двадцать две таблицы.

Формализованное описание объектов предметной области представлено в таблицах 3.1- 3.18.

Таблица «users» содержит информацию о пользователях.

Таблица 3.1 – Структура таблицы «users»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id\_user | INT(8) | Заполняется автоматически | Ключевое поле |
| login | VARCHAR(255) | Заполняется пользователем | Логин пользователя |
| hash\_password | TEXB | Генерируется алгоритмом шифрования | Зашифрованный пароль пользователя |
| hash\_salt | TEXT | Генерируется алгоритмом шифрования | Индивидуальный ключ для пароля пользователя |
| date\_registration | DATE | Заполняется автоматически | Дата регистрации пользователя |
| active\_hash | TEXT | Заполняется автоматически | Активный ключ хеш, служит для восстановления пароля |
| title\_state | TEXT | Заполняется пользователем или автоматически | Причина блокировки пользователя |
| state | BOOL | Заполняется автоматически | Статус пользователя, активен или заблокирован |

Таблица «profiles» содержит подробную информацию о пользователях.

Таблица 3.2 – Структура таблицы «profiles»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id\_profile | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_user | INT(8) | Заполняется автоматически | Код пользователя |
| name | VARCHAR(255) | Заполняется пользователем | Имя пользователя |
| surname | VARCHAR(255) | Заполняется пользователем | Фамилия пользователя |
| date\_birth | DATE | Заполняется пользователем | Дата рождения пользователя |
| id\_image | INT(8) | Заполняется автоматически | Код картинки пользователя |

Таблица «level\_access» содержит информацию об уровнях доступа.

Таблица 3.3 – Структура таблицы «level\_access»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id\_level\_access | INT(8) | Заполняется автоматически | Ключевое поле |
| name | VARCHAR(255) | Заполняется пользователем | Название уровня доступа |
| title | TEXT | Заполняется пользователем | Заголовок уровня доступа |
| description | TEXT | Заполняется пользователем | Описание уровня доступа |
| state | BOOL | Заполняется пользователем | Статус уровня доступа |

Таблица «users\_level\_access» содержит информацию об уровнях доступа.

Таблица 3.4 – Структура таблицы «users\_level\_access»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id\_user\_access | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_user | INT(8) | Заполняется пользователем | Код пользователя |
| id\_level\_access | INT(8) | Заполняется пользователем | Код уровня доступа |

Таблица «modules\_level\_access» содержит информацию об уровнях доступа.

Таблица 3.5 – Структура таблицы «modules\_level\_access»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id\_module\_access | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_module | INT(8) | Заполняется пользователем | Код модуля |
| id\_level\_access | INT(8) | Заполняется пользователем | Код уровня доступа |

Таблица «images\_base» содержит информацию о картинках.

Таблица 3.6 – Структура таблицы «images\_base»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_user | INT(8) | Заполняется автоматически | Код пользователя |
| origin\_name | TEXT | Заполняется автоматически | Оригинальное название картинки |
| hash\_name | TEXT | Заполняется автоматически | Хеш названия картинки |
| path | VARCHAR(255) | Заполняется автоматически | Ссылка на картинку |
| date\_load | DATE | Заполняется автоматически | Дата загрузки картинки |
| state | BOOL | Заполняется пользователем | Статус |

Таблица «load\_images» содержит информацию о загрузках картинкок.

Таблица 3.7 – Структура таблицы «load\_images»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_image | INT(8) | Заполняется автоматически | Код картинки |
| id\_user | INT(8) | Заполняется автоматически | Код зарегистрированного пользователя |
| options\_user | JSON | Заполняется автоматически | Опциональные настройки пользователя |
| date\_load | DATE | Заполняется автоматически | Дата просмотра |

Таблица «user\_images» содержит информацию о картинках который задействованы в профиле пользователя.

Таблица 3.8 – Структура таблицы «user\_images»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |

|  |  |  |  |
| --- | --- | --- | --- |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_image | INT(8) | Заполняется автоматически | Код картинки |
| id\_user | INT(8) | Заполняется автоматически | Код пользователя |
| options | JSON | Заполняется пользователем | опции используемой картинки |

Таблица «files\_base» содержит информацию о файлах.

Таблица 3.9 – Структура таблицы «files\_base»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_user | INT(8) | Заполняется автоматически | Код пользователя |

Продолжение таблицы 3.9

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| origin\_name | TEXT | Заполняется автоматически | Оригинальное название картинки |
| hash\_name | TEXT | Заполняется автоматически | Хеш названия картинки |
| path | VARCHAR(255) | Заполняется автоматически | Ссылка на картинку |
| date\_load | DATE | Заполняется автоматически | Дата загрузки картинки |
| state | BOOL | Заполняется пользователем | Статус |

Таблица «load\_files» содержит информацию о загрузках файлов.

Таблица 3.10 – Структура таблицы «load\_files»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_file | INT(8) | Заполняется автоматически | Код картинки |
| id\_user | INT(8) | Заполняется автоматически | Код зарегистрированного пользователя |
| options\_user | JSON | Заполняется автоматически | Опциональные настройки пользователя |
| date\_load | DATE | Заполняется автоматически | Дата просмотра |

Таблица «user\_files» содержит информацию о файлах который задействованы в профиле пользователя.

Таблица 3.11 – Структура таблицы «user\_images»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |

|  |  |  |  |
| --- | --- | --- | --- |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_file | INT(8) | Заполняется автоматически | Код файла |

Продолжение таблицы 3.11

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id\_user | INT(8) | Заполняется автоматически | Код пользователя |
| options | JSON | Заполняется пользователем | опции используемой файла |

Таблица «type\_variables» содержит информацию о типах переменных.

Таблица 3.12 – Структура таблицы «type\_variables»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| name | VARCHAR(255) | Заполняется пользователем | Название типа переменной |
| title | VARCHAR(255) | Заполняется пользователем | Заголовок типа переменной |
| description | TEXT | Заполняется пользователем | Описание типа переменной |
| state | BOOL | Заполняется пользователем | Статус типа переменной |

Таблица «schema\_variables» содержит информацию о схемах переменных.

Таблица 3.13 – Структура таблицы «schema\_variables»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_type\_variable | INT(8) | Заполняется автоматически | Код типа переменной |
| name | VARCHAR(255) | Заполняется пользователем | Название схемы переменной |
| description | TEXT | Заполняется пользователем | Описание схемы переменной |
| schema | JSON | Заполняется пользователем | Схема переменной |
| state | BOOL | Заполняется пользователем | Статус типа переменной |

Таблица «variables» содержит информацию о переменных.

Таблица 3.14 – Структура таблицы «variables»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_schema\_variable | INT(8) | Заполняется автоматически | Код схемы переменной |
| name | VARCHAR(255) | Заполняется пользователем | Название переменной |
| title | TEXT | Заполняется пользователем | Заголовок переменной |
| description | TEXT | Заполняется пользователем | Описание переменной |
| content | JSON | Заполняется пользователем | Содержимое переменной |
| state | BOOL | Заполняется пользователем | Статус переменной |

Таблица «type\_modules» содержит информацию о типах переменных.

Таблица 3.15 – Структура таблицы «type\_modules»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| name | VARCHAR(255) | Заполняется пользователем | Название типа модуля |
| title | TEXT | Заполняется пользователем | Заголовок типа модуля |
| description | TEXT | Заполняется пользователем | Описание типа модуля |
| state | BOOL | Заполняется пользователем | Статус типа модуля |

Таблица «modules» содержит информацию о модулях.

Таблица 3.16 – Структура таблицы «modules»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id\_module | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_type\_module | INT(8) | Заполняется пользователем | Код типа модуля |
| name | VARCHAR(255) | Заполняется пользователем | Название модуля |
| title | VARCHAR(255) | Заполняется пользователем | Заголовок модуля |
| description | VARCHAR(255) | Заполняется пользователем | Описание модуля |
| lay\_out | JSON | Заполняется пользователем | Шаблон модуля |
| schema\_lay\_out | JSON | Заполняется пользователем | Схема шаблона модуля |
| settings | JSON | Заполняется пользователем | Настройки модуля |
| state | BOOL | Заполняется пользователем | Статус модуля |

Таблица «rows\_modules» содержит информацию о строках в модулях.

Таблица 3.17 – Структура таблицы «rows\_modules»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id\_row\_module | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_module | INT(8) | Заполняется автоматически | Код модуля |
| date\_create | DATE | Заполняется автоматически | Код переменной |
| date\_update | DATE | Заполняется автоматически | Содержание переменной |
| id\_user | INT(8) | Заполняется автоматически | Код пользователя |
| state | BOOL | Заполняется пользователем | Статус переменной |

Таблица «content\_rows» содержит информацию данных в строке модуля.

Таблица 3.18 – Структура таблицы «content\_rows»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_row\_module | INT(8) | Заполняется автоматически | Код строки модуля |
| id\_variable | INT(8) | Заполняется автоматически | Код переменной |
| data | JSON | Заполняется пользователем | Содержание переменной |

В ходе выполнения технического проектирования ПС было разработана диаграмма развёртывания, которая описывает архитектуру развёрнутого программного средства на сервере. Также выбраны и обоснованы решения для реализации программного средства. Разработана модель данных которая описывает структуру используемой базы данных, описаны все таблицы с полями, каждое поле в описании имеет следующие характеристики: название поля, тип данных, источник данных, описании. Также реализована диаграмма сущьность-связь.

4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

4.1 Описание тестируемого стенда

Для проведения тестирования использовался сервер на ОС Ubuntu 18.04.3 LTS x86\_64, характеристики сервера:

* ЦП: Intel(R) Core(TM) i3-4130 CPU @ 3.40GHz;
* ОЗУ: 8 GB;
* ПЗУ: 60 GB SSD;
* подключение к сети Интернет с пропускной способностью: 25/12,5 Мбит/с

4.2 Функциональное тестирование

Для того, чтобы удостоверится в правильности работы программного средства и реализации функциональных требований, было проведено тестирование модулей, которые были разработаны.

В процессе разработки проводилось модульное тестирование, которое помогало выявлять ошибки на ранних этапах, а также позволяли максимально полно проверить все основные модули программного средства.

По мере разработки программного средства, одного модульного тестирования мало и был выбран метод функционального тестирования.

Функциональное тестирование является важным аспектом при тестировании программного средства, так-как каждая функция программы тестируется исходя из поведения реальных пользователей, а также полный проход по всему бизнес сценарию. Поскольку функциональное тестирование не позволяет полностью проверить функцию по всей области её определения, поэтому каждая функция проверяется по основному сценарию работы программного средства [16].

К плюсам данного метода можно отнести:

* обнаружение ошибок интерфейса;
* обнаружение не работающих функций;
* обнаружение некорректного поведения функций;
* имитация поведения реального пользователя;
* корректность внешней структуры данных и отсутствие ошибок.

Из минусов является следующие:

* избыточное тестирование;
* вероятность пропуска логических ошибок.

Функциональное тестирование было реализовано при помощи набора тест-кейсов, собранных в тестовый сценарий, представленный в таблице 4.1

Таблица 4.1 – Результаты функционального тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестовый случай | Ожидаемый результат | Фактический результат |
| 1 | 1. Первое открытие клиентского приложения 2. Регистрация супер-пользователя 3. Успешный вход в аккаунт супер-пользователя | Форма регистрации открыта, пользователь успешно регистрирован, открыта главная страница клиентского приложения. | Совпадает с ожидаемым |
| 2 | 1. Открытие клиентского приложения 2. Открытие формы регистрации 3. Ввод данных 4. Нажать «Регистрация» | Клиентское приложение запущено, форма регистрации открыта, пользователь зарегистрирован, открыта главная страница клиентского приложения. | Совпадает с ожидаемым |
| 3 | 1. Открытие клиентского приложения 2. Открытие формы авторизации 3. Ввод данных 4. Нажать «Вход» | Клиентское приложение запущено, форма авторизации открыта, пользователь авторизован, открыта главная страница клиентского приложения. | Совпадает с ожидаемым |
| 4 | 1. Переход на страницу настроек приложения 2. Редактирование существующей переменной 3. Ввод данных 4. Нажать «сохранить» | Открытие страницы настроек приложения, редактирование существующих переменных, сохранение изменений, переход на страницу настроек. | Совпадает с ожидаемым |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестовый случай | Ожидаемый результат | Фактический результат |
| 5 | 1. Переход на страницу пользователи 2. Выбор необходимого пользователя 3. Блокировка пользователя 4. Ввод причины блокировки 5. Авторизация данными блокированного пользователя 6. Нажать «Вход» | Открытие страницы списка пользователей, открытие необходимого пользователя, блокировка пользователя, ввод причины блокировки, открыть клиентское приложение в другом браузере, авторизоваться под заблокированном пользователе, вывод ошибки пользователь заблокирован и причину блокировки. | Совпадает с ожидаемым |
| 6 | 1. Переход на страницу профиля 2. Редактирование информации 3. Нажать «сохранить» 4. Просмотр информации пользователя другим пользователем | Открытие страницы профиля, редактирование пользовательской информации, данные сохранены, переход на страницу пользователя. | Совпадает с ожидаемым |
| 7 | 1. Переход на страницу модуля 2. Добавление записи модуля 3. Нажать сохранить 4. Просмотреть запись другим пользователем | Переход на страницу модуля, добавление новой записи модуля, запись данных, переход на страницу записи модуля. | Совпадает с ожидаемым |
| 8 | 1. Переход на страницу пользователя 2. Выбрать необходимого пользователя 3. Поднять уровень доступа пользователя 4. Нажать «сохранить» 5. Авторизовать под данным пользователей 6. Проверить уровень доступа | Открыть страницу списка пользователей, выбор необходимого пользователя, поднятие уровня доступа, сохранение изменений, авторизация под данным пользователем, проверка уровня доступа. | Совпадает с ожидаемым |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестовый случай | Ожидаемый результат | Фактический результат |
| 9 | 1. Переход на страницу файлов 2. Добавление нового файла 3. Нажать «Сохранить» 4. Проверить доступность файла | Открытие страницу файлов, загрузить файл, проверка доступности файла. | Совпадает с ожидаемым |
| 10 | 1. Переход на не существующую страницу 2. Вывод ошибки «404: страница не найдена» | Ввести в адрес клиентского приложения не существующую ссылку на страницу, вывод ошибки на экран | Совпадает с ожидаемым |
| 11 | 1. Переход на страницу авторизации 2. Ввести не правильны пароль 3. Нажать «войти» 4. Повторить 2 и 3 пункт 3 раза 5. Вывод ошибки пользователь заблокирован | Открыть страницу авторизации, провести 3 попытки авторизации с не правильным паролем, вывод ошибки на экран «пользователь заблокирован» | Совпадает с ожидаемым |
| 12 | 1. Переход на страницу метки 2. Добавить новую метку 3. Заполнить все поля кроме названия 4. Нажать «добавить» 5. Вывод ошибки проверки данных | Открыть страницу метки, заполнить все поля кроме поля название, нажать «добавит», вывод ошибки проверки данных | Совпадает с ожидаемым |
| 13 | 1. Переход на страницу к которой нет прав доступа 2. Вывод ошибки «нет прав доступа» | Открыть страницу к которой у данного пользователя нет прав доступа, вывод на экран ошибки «У вас нет доступа к данной странице» | Совпадает с ожидаемым |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестовый случай | Ожидаемый результат | Фактический результат |
| 14 | 1. Переход на несуществующую страницу за счёт ввода в адрес клиентского приложения несуществующей страницы 2. Вывод ошибки «Страница не найдена» | Открыть клиентское приложение, ввести в адресную строку несуществующую страницу, вывод на экран ошибки «страница не найдена» | Совпадает с ожидаемым |
| 15 | 1. Переход на страницу иконки 2. Выбор необходимой категории иконок 3. Ввести в поиск название искомой иконки 4. Выбрать нужную иконку 5. Скопировать кодовое название иконки и использовать для настроек модуля 6. Переход на страницу настроек модулей 7. Выбор необходимого модуля 8. Изменение иконки модуля 9. Нажать «Сохранить» | Открыть страницу иконок, выбор необходимой категории, ввести в поиск ключевое слово, нажать на необходимую иконку, скопировать кодовое название, перейти на страницу настроек необходимого модуля, в настройках модуля вставить кодовое название иконки в соответствующее поле, сохранить, изменить иконку модуля в панели меню. | Совпадает с ожидаемым |
| 16 | 1. Переход на страницу компонентов 2. Выбор необходимого типа переменно 3. Просмотр как работает переменная | Переход на страницу компонентов, выбор необходимого типа переменной, просмотреть как работает переменная. | Совпадает с ожидаемым |
| 17 | 1. Переход на страницу профиля 2. Нажать «Редактировать» 3. Изменить пользовательские данные 4. Нажать «Сохранить» 5. Проверить изменились ли данные пользователя | Переход на страницу профиля, переход в режим редактирования, редактирование данных профиля пользователя, сохранение данных, проверка изменились ли данные на странице пользователя. | Совпадает с ожидаемым |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестовый случай | Ожидаемый результат | Фактический результат |
| 18 | 1. Переход на страницу настроек 2. Открыть настройки внешнего клиентского приложения 3. Изменить настройки прокси сервера для доступа к API 4. Нажать «Сохранить» 5. Проверить доступность API из внешнего приложения | Переход на страницу настроек, открыть страницу настроек внешнего клиентского приложения, изменить настройки прокси сервера, проверка доступа к API из внешнего клиентского приложения | Совпадает с ожидаемым |

4.3 Тестирование производительности

Тестирование производительности – это комплекс типов тестирования, целью которого является определение работоспособности, стабильности, потребления ресурсов и других атрибутов качества приложения в условиях различных сценариев использования и нагрузок. Тестирование производительности позволяет находить возможные уязвимости и недостатки в системе с целью предотвратить их пагубное влияние на работу программы в условиях использования. Необходимые параметры работы системы в определенной среде можно тестировать с помощью:

* определения рабочего количества пользователей приложения;
* измерение времени выполнения различных операций системы;
* определения производительности приложения при различных степенях нагрузки;
* определения допустимых границ производительности программы при разных уровнях нагрузки [17].

Нагрузочное тестирование – тестирование времени отклика приложения на запросы различных типов, с целью удостовериться, что приложение работает в соответствии с требованиями при обычной пользовательской нагрузке [18].

Для проведения тестирования реализован тестовый чат, к которому будут подключены виртуальные пользователи. С помощью утилиты Apache Benchmark с другой машины подключаются несколько Постоянных HTTP-соединений клиентов (время тестирования 20 секунд). В это время сторонней формой начинается запись в БД сообщения. Для каждого числа Постоянное HTTP-соединение соединений настроено по 5 итераций, полученные данные усредняем и полученный результат внесен в таблицу 4.2.

Таблица 4.2 - Результаты нагрузочного тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество запросов | Запросов в секунду | Среднее время на запрос в ms (по всем одновременным запросам) | Минимальное время выполнения запроса в ms | Максимальное время выполнения запроса в ms |
| 10 пользователей | | | | |
| 1285 | 64 | 15.618 | 16 | 9064 |
| 100 пользователей | | | | |
| 1284 | 64.12 | 15.765 | 17 | 13347 |
| 500 пользователей | | | | |
| 1236 | 64.71 | 16.611 | 23 | 16078 |
| 1000 пользователей | | | | |
| 1528 | 75.02 | 13.354 | 17 | 16785 |

4.4 Примеры ошибок

В результате тестирования программного средства было выявлено несколько ошибок в работе API, все ошибки на стороне клиентского приложения обработаны и выводятся соответствующие уведомления.

Ниже представлены часто возникающие ошибки на стороне клиента:

Ошибка, возникающая при не правильном вводе данных или не заполнении обязательных полей, ошибка представлена на рисунке 4.1

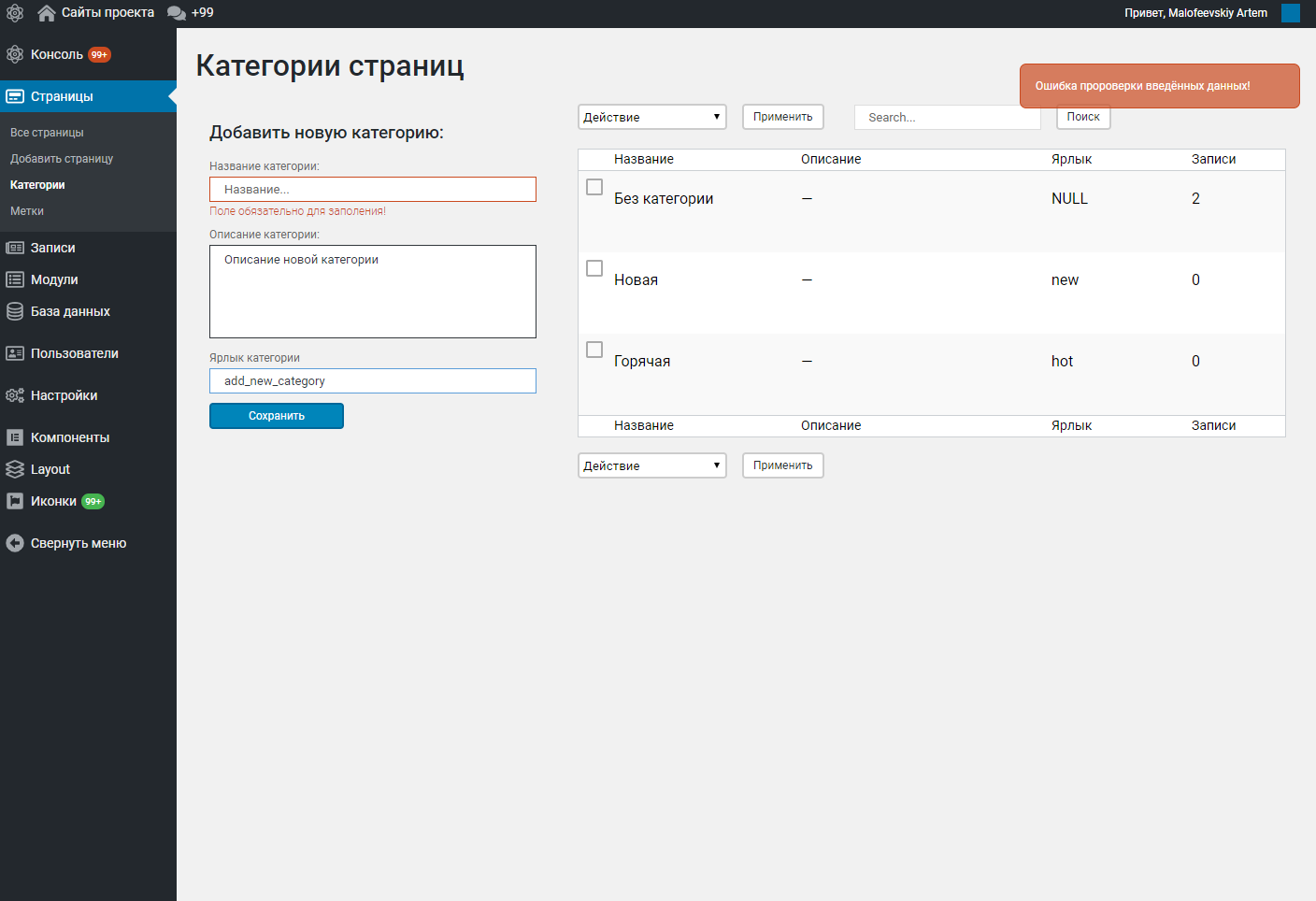


Рисунок 4.1 – Ошибка проверки введённых данных

Ошибка, возникающая если пользователь пытается открыть раздел, к которому у его нет доступа, ошибка представлена на рисунке 4.2.

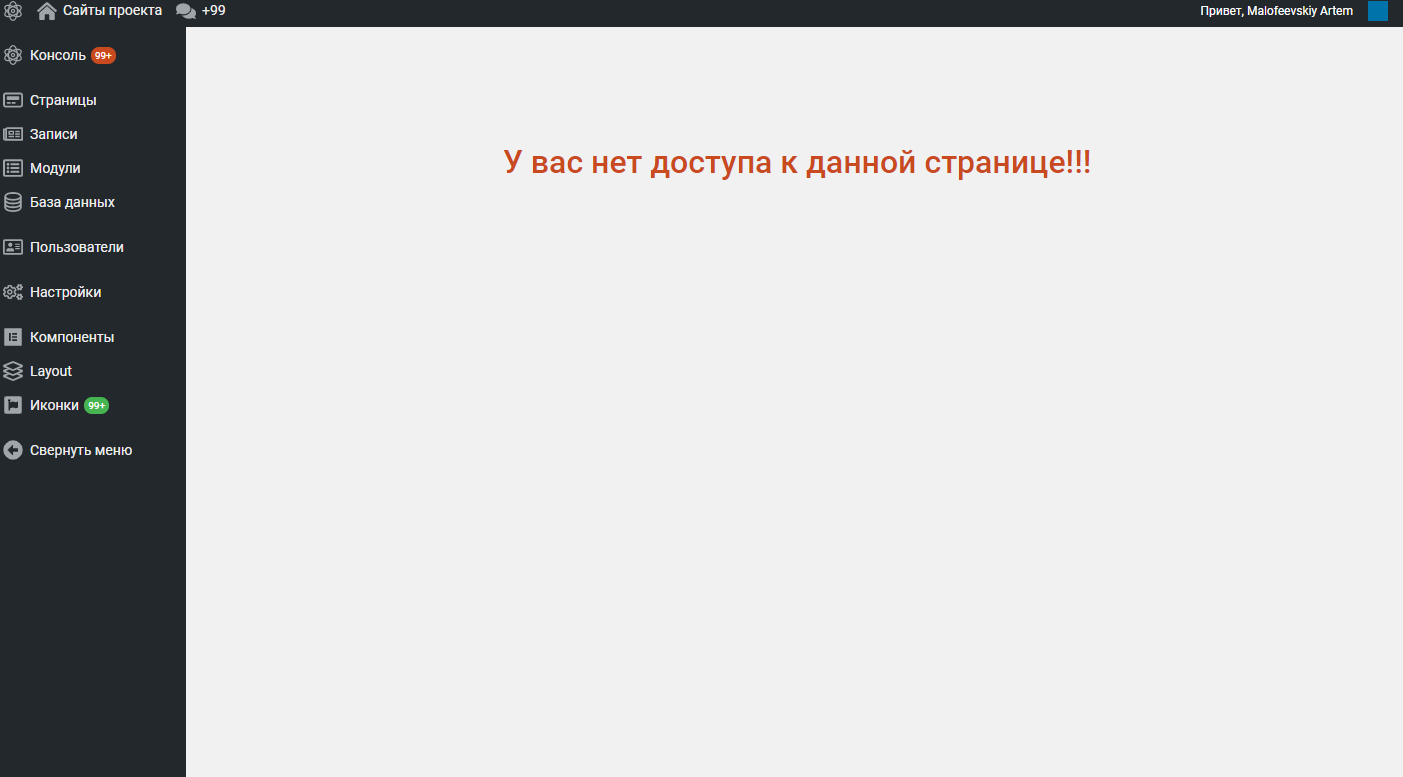


Рисунок 4.2 – Ошибка прав доступа к странице

Ошибка, возникающая если пользователь заблокирован, пользователь может быть заблокирован в 2х случаях если его заблокирует пользователь с уровнем выше данного пользователя или заблокирует система если будет введено более 3х попыток ввода пароля, ошибка представлена на рисунке 4.3.

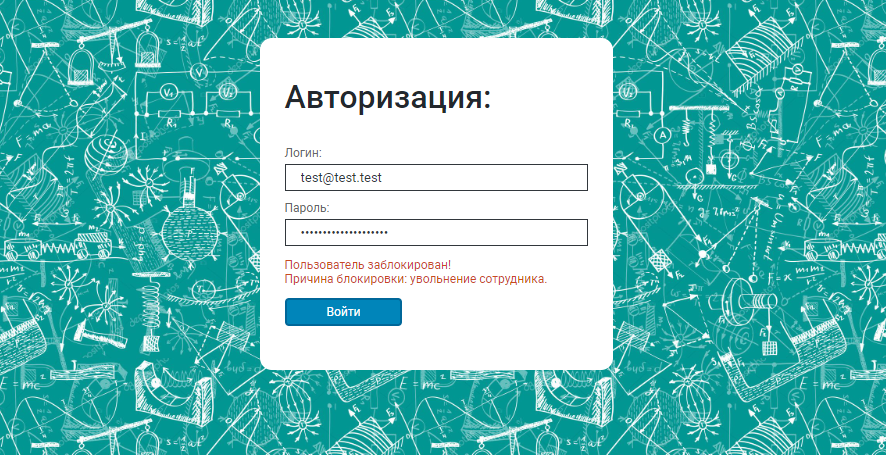


Рисунок 4.3 – Ошибка пользователь заблокирован

Ошибка, возникающая если пользователь пытается перейти на не существующую страницу, текст ошибки «404: страница не найдена!», ошибка представлена на рисунке 4.4. Если страница ранее существовала, но ее удалили, то в тексте ошибки будет «Страница уделена!», данная ошибка представлена на рисунке 4.4.

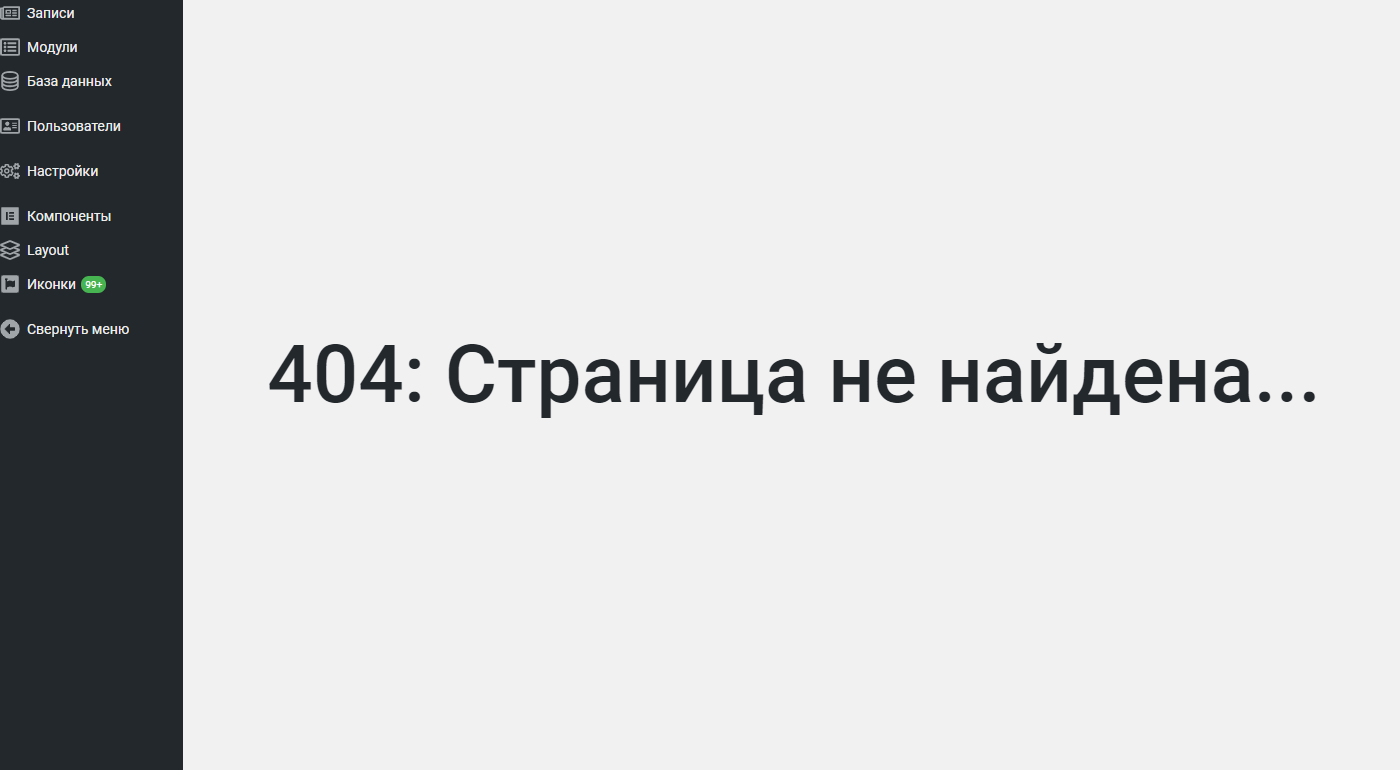


Рисунок 4.4 – Ошибка страница не найдена

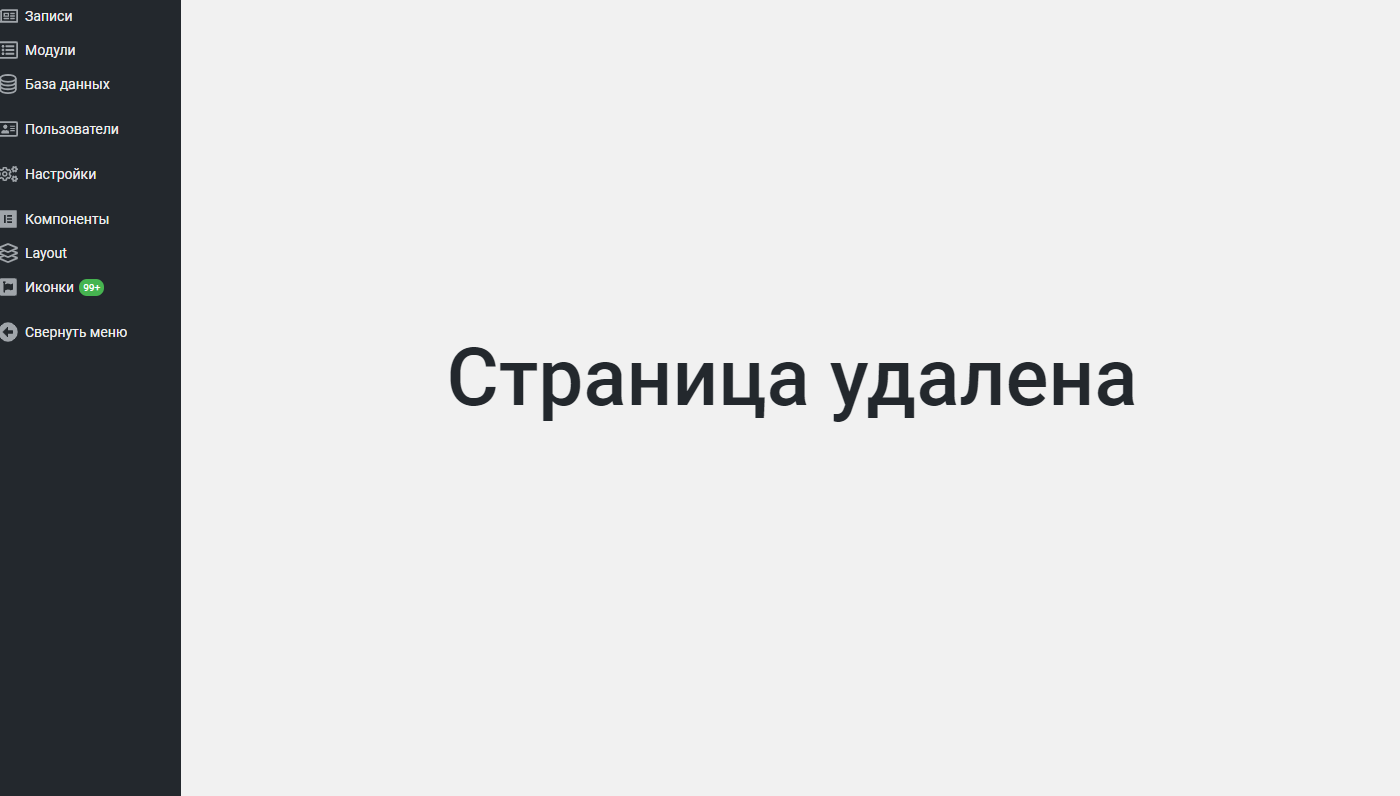


Рисунок 4.5 – Ошибка страница удалена

Ошибка, возникающая если сервер, не обработал запрос, превышен интервал ожидания, или другая иная проблема будет выведена ошибка с текстом «Проблемы на стороне сервера», ошибка представлена на рисунке 4.6.

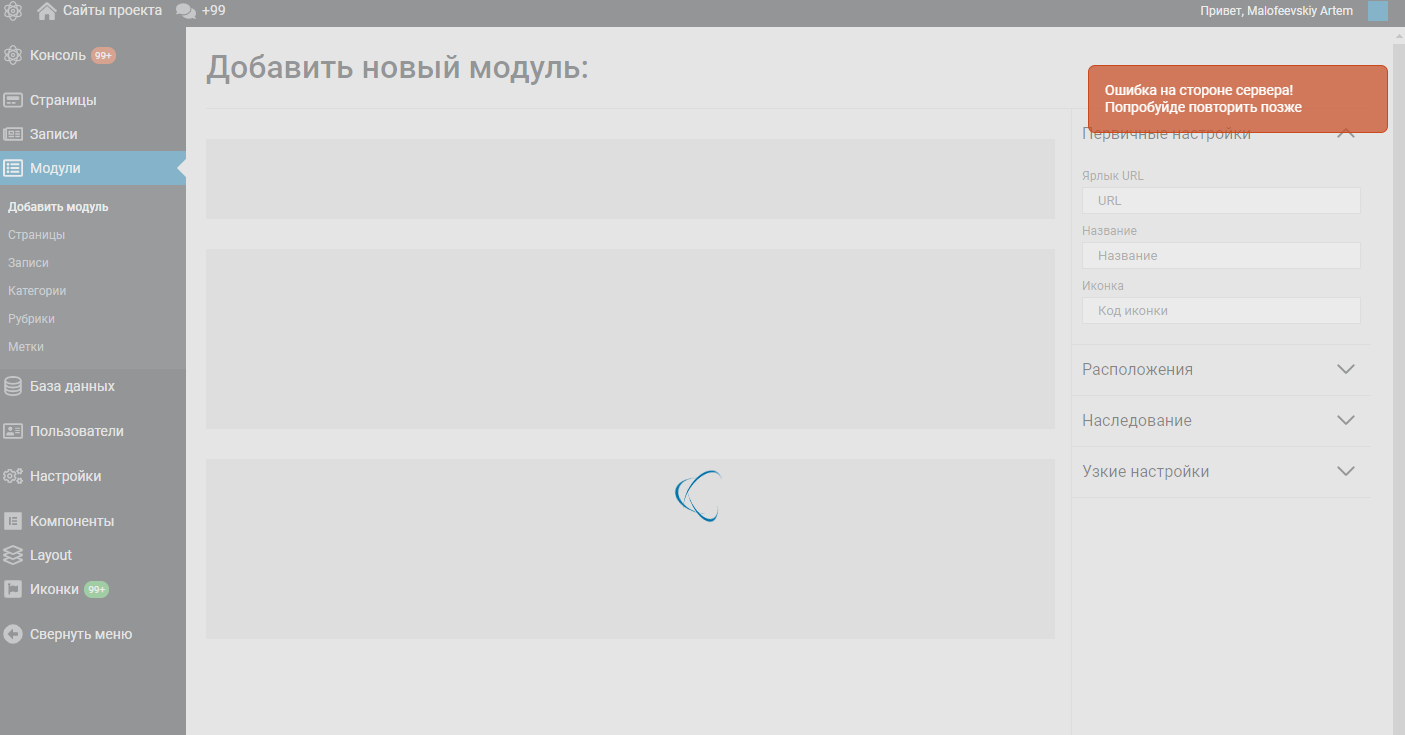


Рисунок 4.6 – Ошибка на стороне сервера

Ошибка, возникающая если пользователь не авторизован, но пытается открыть страницу по ссылке, ошибка отображена на рисунке 4.7.

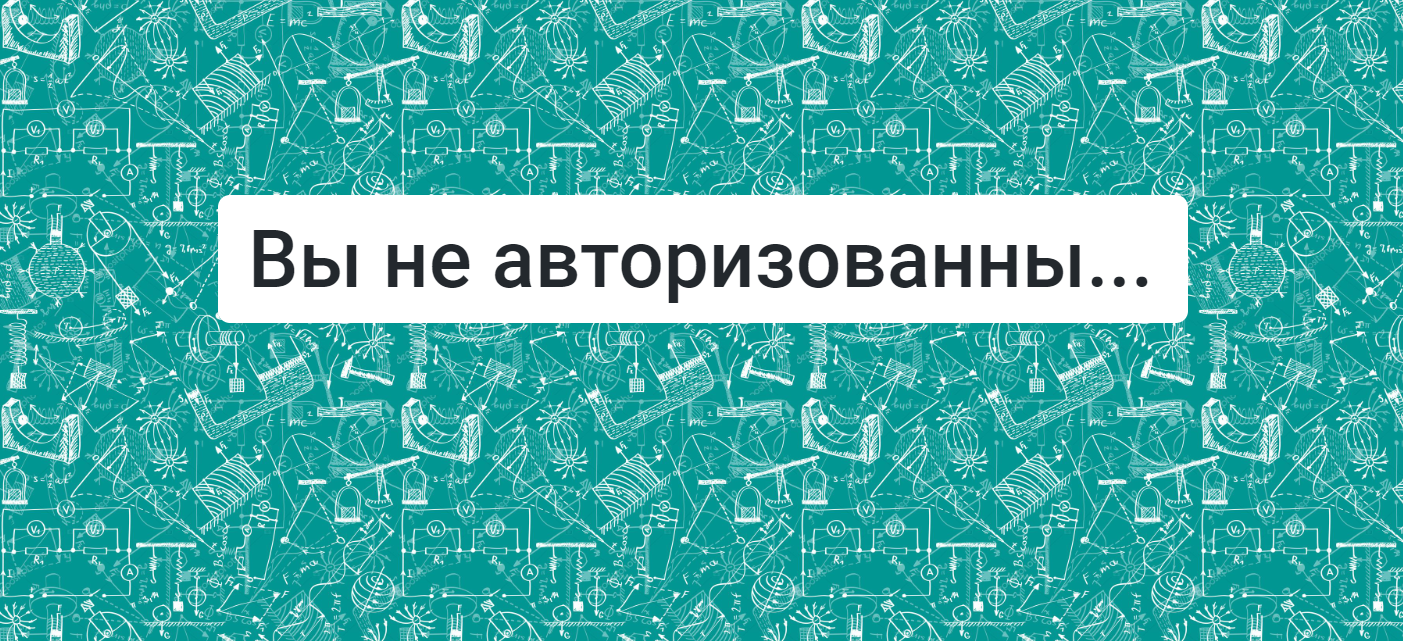


Рисунок 4.7 – Ошибка, не авторизованный пользователь

Ошибка, возникающая в случае неправильной компиляции динамических страниц, ошибка представлена на рисунке 4.8.

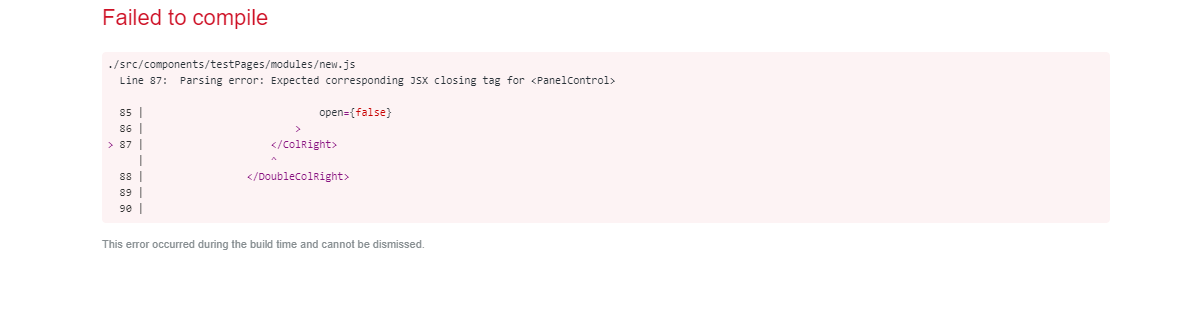


Рисунок 4.8 – Ошибка компиляции динамических страниц

Ошибка, возникающая при не верном API запросе с внешнего клиента к серверу, ошибка представлена на рисунке 4.9.

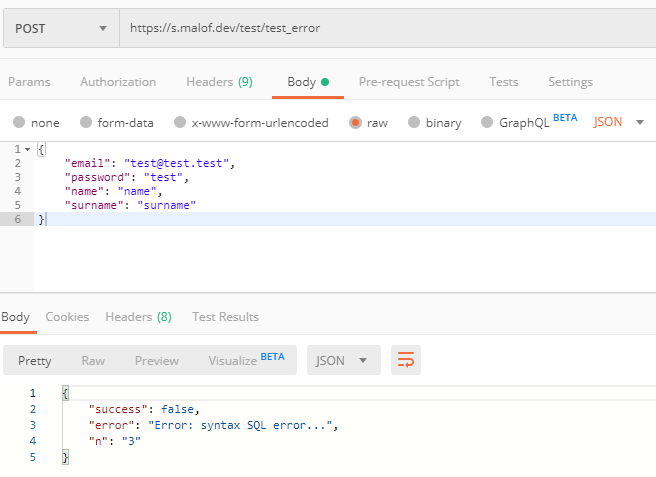


Рисунок 4.9 – Ошибка API из внешнего клиентского приложения

В результате проведённого тестирования программного средства были использованы методы функционального тестирования и тестирование производительности. Были описаны все используемые тест-кейсы в функциональном тестировании, также представлена таблица с результатами нагрузочного тестирования. Было описано большинство возникающих ошибок при работе с клиентским приложением.

5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для использования программного средства необходимо использовать выделенный или виртуальный сервер, операционная система может использоваться любая, но рекомендуется Linux Ubuntu 18.04.3 LTS x86\_64. Рекомендованные системные параметры:

* ЦП: Intel(R) Core(TM) i3-4130 CPU @ 3.40GHz;
* ОЗУ: 8 GB;
* ПЗУ: 60 GB SSD;
* подключение к сети Интернет с пропускной способностью: 25/12,5 Мбит/с.

На сервере должен быть настроен apache2 или nugix, также установлен Node JS/

Перед запуском приложения необходимо настроить конфигурационный файл приложения:

* JWT\_SECRET – секретный ключ для хеширования JWT;
* EXP\_JWT\_HOURS – время которое будет активен JWT в часах;
* SALT\_SEVRET – секретный ключ хеширования соли;
* DATA\_BASE – данные для подключения к БД.

Поле чего можно приступить к запуску приложения, для запуска можно выбрать несколько конфигураций:

1. Запуск приложение с единым сервисом, команда для запуска npm run one.
2. Запуск приложения с 2 микро-сервисами:

* Первый микро-сервис работает с пользователями и файлами;
* Второй микро-сервис работает с модулями

Команда для запуска npm run double.

1. Запуск приложения с 4 микро-сервисами

* Первый микро-сервис работает с пользователями;
* Второй микро-сервис работает с файлами;
* Третий микро-сервис работает предустановленными модулями;
* Четверной микро-сервис работает с индивидуальными модулями.

Команда для запуска npm run full.

После запуска серверного приложения необходимо запустить клиентское приложения для администрирования, приложения запускается с помощью команды npm run client.

После выполнения данных команд в сети будет доступен клиент, с помощью браузера на его можно будет перейти, ссылка на клиентское приложение устанавливается в настройках apache2.

При первом входе в приложение необходимо зарегистрировать супер-пользователя. После регистрации можно начитать работать с приложением, добавлять новых пользователей, работать с группами доступа, загружать файлы, работать с модулями.

Изначально в системе присутствуют предустановленные модули:

* страницы;
* категории страниц;
* записи;
* рубрики записей;
* метки для страниц и записей;
* файлы.

Модуль страниц представлен из 6 переменных которые можно изменять или добавлять новые, на рисунке 5.1 представлен модуль страниц со списком переменных.

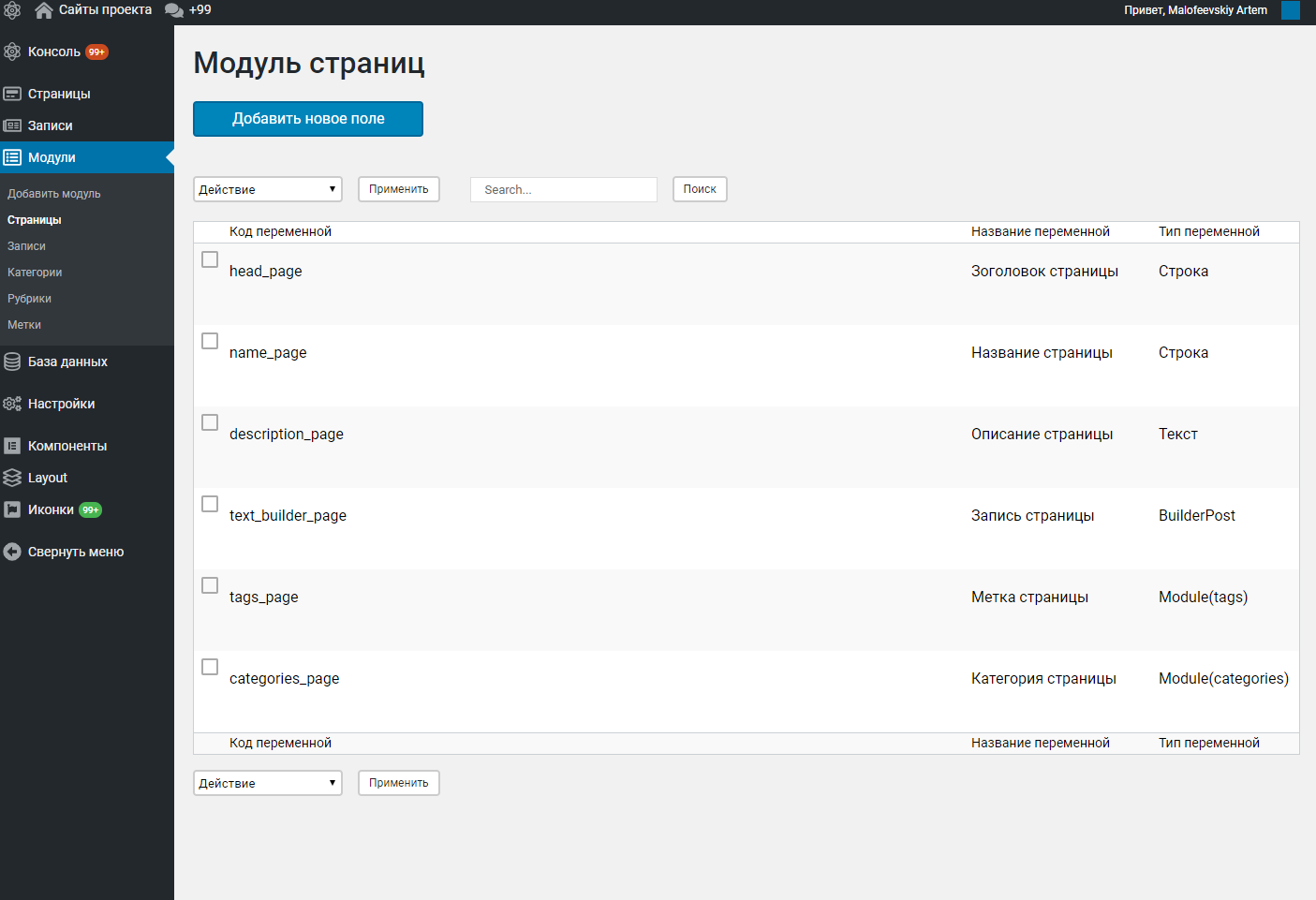


Рисунок 5.1 – Список переменных модуля страниц

Первые 3 переменные являются стандартными текстовыми, четвёртая переменная является билдером записи с возможность вставки HTML кода. Также используются две не стандартные переменные, тип данных переменных является модулем, а именно теги и категории. Для данных переменных отдельно предусмотрены модули тегов и категорий. Данные модули содержат в себе 4 переменные: название, описание, ярлык и записи. Поле ярлык используется для фильтрации записей, а поле записи используется для автоматического подсчёта сколько раз было использована категория. На рисунке 5.2 представлен модуль категории.

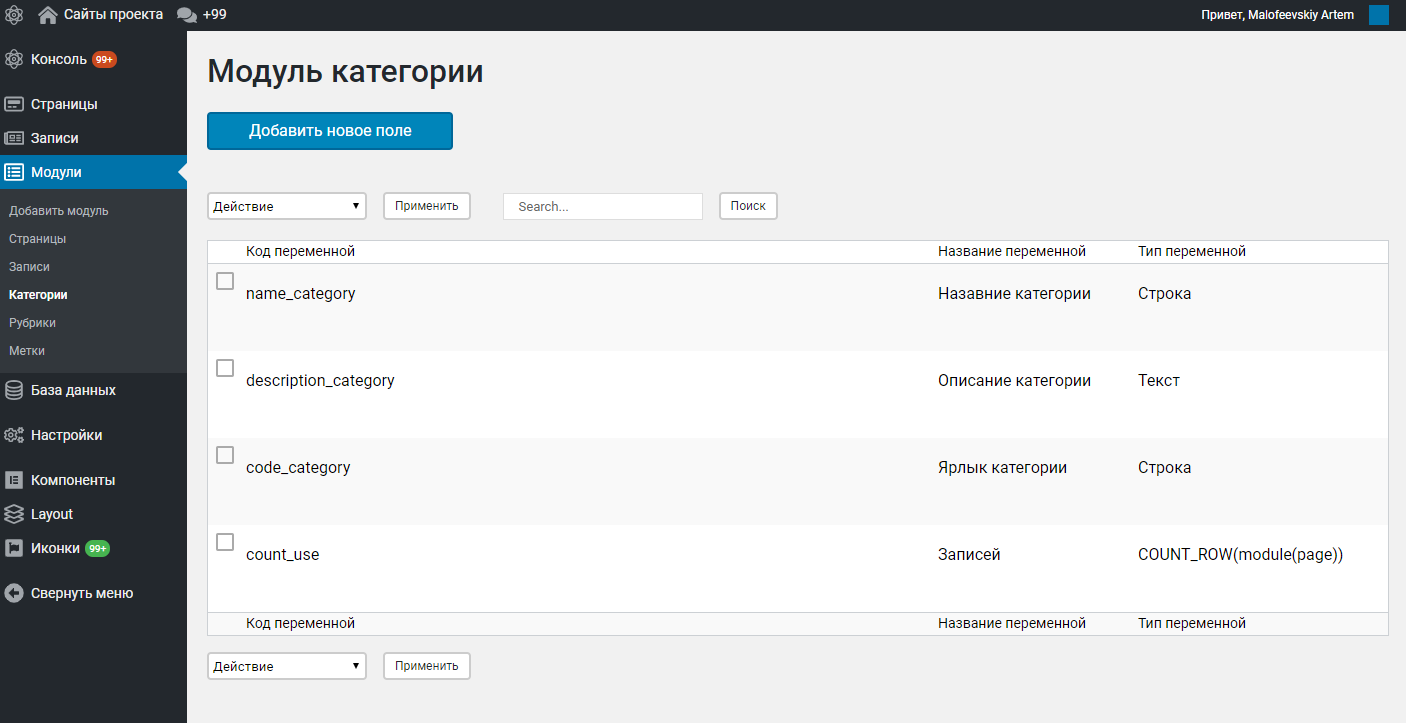


Рисунок 5.2 – Список переменных модуля категорий

6 ОПРЕДЕЛЕНИЕ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ

# 6.1 Характеристики программного продукта

Программное средство для генерации индивидуальных модулей и API предназначено для упрощенного и быстрого развёртывания микро-сервисного приложения на стороне сервера, приложение обладает следующими базовыми функциями:

* регистрация и авторизация пользователей;
* надёжное шифрование паролей;
* создание сессии для пользователей с использованием JWT;
* защита сессии от не санкционированного доступа из вне;
* пользовательская настройка прав доступа к админской и внешней части приложения;
* выдача прав доступа пользователям;
* защита CROS от не санкционированного доступа к API проекта;
* аналитика просмотра контента уникальными пользователями;
* хранение и рассылка почтовых шаблонов(оповещений) по электронной почте;
* аналитика по рассылке;
* создание индивидуальных модулей;
* ограничение доступа к пользовательскому контенту;
* обработка и хранение файлов на сервере;
* сервисы, работающие в реальном времени.

# 6.2 Расчёт стоимостной оценки затрат программного продукта

Рассчитаем основную заработную плату исполнителя модернизируемого программного средства.

1. Основная заработная плата рассчитывается по формуле:

(6.1)

где n – количество исполнителей, n = 1;

– количество часов работы в день, 8 ч.;

– часовая тарифная ставка i-го исполнителя, 5,2 руб.;

– эффективный фонд рабочего времени i-го исполнителя, 96 дней;

– коэффициент премирования, 2,48.

В модернизации будет участвовать один исполнитель (таблица 6.1).

5,2·8·96·2,48 = 9904,13 руб.

Таблица 6.1 – Разряды, ставки и тарифные коэффициенты работников

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование должности | Разряд | Тарифный коэффициент | Часовая тарифная ставка, руб. |
| Инженер-программист | 10 | 2,48 | 5,2 |

1. Дополнительная заработная плата определяется в зависимости от норматива прибавки к заработной плате в процентах по формуле:

, (6.2)

где – дополнительная заработная плата исполнителя ПС,

1980,83 руб.;

– норматив дополнительной заработной платы,

20 %;

– сумма основной заработной платы исполнителя ПС,

9904,13 руб.

1. Отчисления в фонд социальной защиты населения и обязательное страхование определяются в соответствии с действующими законодательными актами по нормативу в процентном соотношении к фонду основной и дополнительной зарплаты исполнителей, определённой по нормативу, установленному в целом по организации. Вычисляется по формуле:

, (6.3)

где – сумма основной заработной платы исполнителя ПС, ;

– дополнительная заработная плата исполнителя ПС, .;

– норматив отчислений в фонд социальной защиты населения и обязательное страхование, 35 %.

1. Расходы по статье «Машинное время» включают оплату машинного времени, необходимого для модернизации и отладки ПС, которое определяется по нормативам на 100 строк LOC (Нмз) машинного времени в зависимости от характера решаемых задач и типа ПК. Рассчитывается по формуле:

, (6.4)

где – цена одного машино-часа, 0,098 руб.;

– общий объём, 25822 LOC;

– норматив расхода машинного времени на отладку 100 строк LOC,3%.

1. Расходы по статье «Прочие затраты» включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по научной организации, в процентах к основной заработной плате.

, (6.5)

где – сумма основной заработной платы исполнителя ПС, 9904,13 руб.;

– норматив прочих затрат,  20 %.

руб.

1. Расходы по статье «Накладные расходы», связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды, относятся по нормативу (Ннр) в процентном отношении к основной заработной плате исполнителя. Рассчитывается по формуле:

, (6.6)

где – сумма основной заработной платы исполнителя ПС, ;

– норматив накладных расходов в целом по научной организации, 58 %.

1. Общая сумма производственной себестоимости (Спр) на ПС рассчитывается по формуле:

, (6.7)

где – сумма основной заработной платы исполнителя ПС,

;

– дополнительная заработная плата исполнителя ПС,;

ЗСОЦ – отчисления в фонд социальной защиты населения, ЗСОЦ = руб.;

РПЗ – расходы по статье «Прочие затраты», РПЗ = 1980,83 руб.;

расходы по статье «Машинное время», ;

– расходы по статье «Накладные расходы»,

1. Расходы пользователя на оплату услуг по сопровождению и адаптацию ПС, которые определяются по нормативу (Нс) и по формуле:

, (6.8)

где – норматив расходов на сопровождение и адаптацию, 20%;

– общая сумма производственной себестоимости на ПС,

1. Полная себестоимость ПС определяется по формуле:

, (6.9)

где СПР – общая сумма производственной себестоимости на ПС, ;

РС – расходы пользователя на оплату услуг по сопровождению и адаптацию ПС, РС = руб.

1. Прирост прибыли за счёт экономии расходов, связанный с высвобождением работника с повременной оплатой труда, определяется по формуле:

 (6.10)

где – коэффициент премий за выполнение плановых заданий, ;

– абсолютное высвобождение работников, = 1;

– заработная плата высвобождаемых работников i-ой категории, З = 915,2 руб.;

– процент дополнительной заработной платы, 20%;

– ставка отчислений от заработной платы, включаемых в себестоимость продукции, 35%;

– количество категорий, которым принадлежат высвобожденные работники.

В результате полученных расчётов можно посчитать экономическую эффективность ПС. За счёт прироста прибыли с экономии расходов мы получаем за год 15387,87 чистой прибыли.

Таким образом, при определении экономической эффективности использования программного средство для генерации индивидуальных модулей и API:

* полная себестоимость ПС в реализации проекта составляет СП = руб.;
* чистая прибыль Пч = 15203,65 руб.;

Таким образом, процент рентабельности разработки программного средство для генерации индивидуальных модулей и API является 52.91% что является экономически выгодным.

ЗАКЛЮЧЕНИЕ

За время прохождения преддипломной практики было разработано программное средство генератор API на платформе Node JS, позволяющие быстро развёртывать клиент-серверное приложение, использовать предустановленные модули, создавать индивидуальные модули, масштабировать приложение.

При реализации проекта была изучена методика проектирования и разработки веб-сервисов используя Spring Boot. Проведено исследование аналогов программного средства и были выявлены их достоинства и недостатки. На основе полученной информации были сформулированы требования к собственному программному средству.

Выполнен анализ предметной области. В процессе работы и проектирования выполнены следующие исследования и разработки:

– созданы модели IDEF0;

– построена и проработана модель вариантов использования;

– построена схема базы данных и модель «Сущность-связь»;

Проведено функциональное проектирование, выявлены все связи и особенности взаимодействия.

В ходе технического проектирования на основе функциональных моделей была разработана диаграмма развёртывания, построены алгоритмы.

Проведено тестирование основных модулей программного средства, проверены функции расчета. Разработана методика работы с программным средством для упрощения работы пользователя с ней.

Был проведён расчёт экономического эффекта от внедрения программного средства, который показал его целесообразность.

Разработанное программное средство обеспечивает решение поставленных перед ним функциональных задач.

Таким образом, цель дипломного проекта достигнута.

Данное программное приложение может быть дополнено и модернизировано.

CПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] The State of REST in 2019 [Электронный ресурс] – 2019 – Режим доступа: https://medium.com/javascript-in-plain-english/the-state-of-rest-in-2019-75005eaf05b9

[2] Современный мир держится на API [Электронный ресурс] – 2019 – Режим доступа: https://habr.com/ru/company/softwareag/blog/449544/

[3] Антиплагиат [Электронный ресурс]. – 2019. – Режим доступа: https://users.antiplagiat.ru/. Дата доступа: 25.12.2019.

[4] приложения, реализованные с использованием микро-сервисов и API [Электронный ресурс] – 2019 – Режим доступа: https://github.com/search?q=microservices+api

[5] Airship is a framework for Node.JS [Электронный ресурс] – 2019 – Режим доступа: https://github.com/Naltox/airship

[6] Управление хранить бесплатно с программным обеспечением Hiboutik POS [Электронный ресурс] – 2019 – Режим доступа https://www.hiboutik.com/ru/

[7] amoCRM – "Хотите увеличить продажи?" [Электронный ресурс]. – 2019 – Режим доступа: https://www.amocrm.ru/

[8] Методология IDEF0 [Электронный ресурс] – 2019 – Режим доступа: https://itteach.ru/bpwin/metodologiya-idef0

[9] Бахтизин, В.В. Структурный анализ и моделирование в среде CASE-средства BPWin: Учеб. пособие по курсу “Технология проектирования программ” для студ. спец. 40 01 01. / Глухова Л.А. – Мн.: БГУИР, 2002. – 44с.\

[10] Диаграмма вариантов использования (use case diagram) [Электронный ресурс] – 2019 – Режим доступа: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl4/gl4.html

[11] Node JS [Электронный ресурс] – 2019 – Режим доступа: https://nodejs.org/ru/

[12] React JS [Электронный ресурс] – 2019 – Режим доступа: https://ru.reactjs.org/

[13] PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс] – 2019 – Режим доступа: https://www.postgresql.org/

[14] Форта, Б. SQL за 10 минут. Б. Форта. – М. : Вильямс, 2019. – 288с.

[15] Базы данных и модели данных. Основы проектирования реляционных баз данных [Электронный ресурс] – 2019 – Режим доступа: http://edu.tltsu.ru/sites/sites\_content/site216/html/media67139/theor\_bd.pdf

[16] Тестирование. Фундаментальная теория [Электронный ресурс] – 2016 – Режим доступа: https://habr.com/ru/post/279535/

[17] Блэк, Р. Ключевые процессы тестирования программного обеспечения. Р. Блэк. – М. : Символ, 2012. – 772с.

[18] Тестирование производительности [Электронный ресурс] – 2019 – Режим доступа: https://qalight.com.ua/baza-znaniy/testirovanie-proizvoditelnosti/

ПРИЛОЖЕНИЕ А Исходный код приложения

**(Обязательное)**

**Текст программного средства**